UNIVERSITY OF CALIFORNIA
RIVERSIDE

Visual-Inertial Odometry: Efficiency and Accuracy

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

Doctor of Philosophy

in

Electrical Engineering

by

Xing Zheng

December 2017

Dissertation Committee:

Dr. Anastasios Mourikis, Chairperson
Dr. Amit Roy-Chowdhury
Dr. Jay Farrell

The Dissertation of Xing Zheng is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisor, Prof. Anastasios Mourikis, for his brilliant guidance and continuous support over the past five years. His professional knowledge and rigorous thinking style inspire me on how to be a good researcher and engineer, which will have a long-term influence on my future career. Without his help, I would never be able to finish my Ph.D. program.

I would also like to thank other members in my oral and defense committees, Prof. Amit Roy-Chowdhury, Prof. Christian Shelton, Prof. Jay Farrell and Prof. Qi Zhu, for their valuable suggestions and advices for my research and graduate study. Their insightful ideas from different perspectives always help me a lot.

In addition, I want to thank Dr. Mingyang Li. Mingyang is an expert in visual-inertial localization, and he is always willing to help me when I encounter difficulties, no matter it's a mathematical derivation or coding problem. The knowledge I learned from him and inspiration gained from the discussion with him have always been beneficial to building my professional skills. His passion and hard work in research is a model Ph.D. to me and encouraged me when I felt depressed.

At last, my thanks go to other members at the UCR Control and Robotics Lab, Dr. Dongfang Zheng, Dr. Haiyu Zhang, Dr. Paul Roysdon, Hongsheng Yu, Dr. Sheng Zhao, Shukui Zhang, Xinyue Kan and Dr. Yiming Chen, for the discussion, work, food, play and travel in these years. I would also like to thank my friends at Riverside, Fei Ye, Jingyu Zhou, Luting Yang, Meng Zhao, Shiwen Chen, Shu Zhang, Tan Yu, Xin Li, Yan Zhu and Zening Li, for making my Ph.D. life colorful and joyful.

This work was supported by the National Science Foundation (grant no. IIS-1117957, IIS-1253314 and IIS-1316934), and by Google's project Tango.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Visual-Inertial Odometry:
Efficiency and Accuracy

by

Xing Zheng

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2017
Dr. Anastasios Mourikis, Chairperson

Accurate localization is essential in many applications such as robotics, unmanned aerial vehicles, virtual reality, and augmented reality. In this work, we focus on the localization of a platform in an unknown environment, with an inertial measurement unit (IMU) and a monocular camera. This task is often termed *visual-inertial odometry* (VIO).

In this work, we focus improving the computational efficiency and accuracy of the state of the art in VIO algorithms. Specifically, to improve computational efficiency we first propose the Decoupled Estimate-Error Parameterization (DEEP) that addresses the high dimensionality of the estimation problem. An extended Kalman filter (EKF) VIO algorithm is re-formulated in the DEEP framework, using measurements from a rolling-shutter camera. The DEEP-EKF formulation is evaluated through Monte-Carlo simulations and real-world experiments, which shows substantial computational gains, while incurring only a small loss of estimation performance.

To achieve improved estimation accuracy, we describe three key methods. First, we propose high-fidelity sensor modeling, along with *online* self-calibration. An additional

contribution of the work is the novel method for processing the measurements of the rolling-shutter camera, which employs an approximate representation of the estimation *errors*, instead of the state itself. Both Monte-Carlo simulations and real-world experiments are conducted to demonstrate the improved estimation precision of the proposed approach compared to existing ones.

We also propose a direct VIO algorithm, which utilizes image patches extracted around image features, and formulates measurement residuals in the image intensity space directly. A detailed evaluation of the algorithm demonstrates that the use of photometric residuals results in increased pose estimation accuracy, with approximately 23% lower estimation errors, on average in our testing.

At last, we extend the direct VIO formulation to a semi-dense framework, where all informative areas in images are used. Photometric triangulation and a novel noise model, which accounts for noise during the image formation and interpolation errors, are employed in this work. Through Monte-Carlo simulations and real-world experiments, we demonstrate that the proposed semi-dense VIO outperforms the direct VIO and the point-feature-based method, in terms of the estimation accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Localization with Visual and Inertial Sensors

Localization is the task of estimating the pose (position and orientation) of a moving platform using sensor data. These sensors provide noisy measurements of quantities related to the pose and/or motion of the platform, and pose estimates are generated by processing these measurements. In many applications such as robotics, unmanned aerial vehicles, augmented reality, and virtual reality, the ability to accurately localize is essential and thus has attracted significant research interest.

When GPS signals are available, they can be used to provide accurate localization solutions [24, 111]. However, GPS-based approaches cannot function well in environments where the GPS signal is shadowed, most importantly, in indoor environments. To address this problem, a number of solutions have been proposed, with different combinations of sensors. Generally, these sensors can be divided into two types: *proprioceptive* and *exteroceptive* ones. Proprioceptive sensors capture the ego-motion of the platform itself. Typical

proprioceptive sensors include wheel encoders [10, 105], which measure the linear and rotational velocities, or an inertial measurement unit (IMU) [3], which measures specific force and rotational velocity. Exteroceptive sensors, on the other hand, capture the information of the surroundings. Typical exteroceptive sensors include the laser scanner [1], sonar [18], radar [23] and camera [41, 64].

We here focus on localization based on a camera and an IMU. This choice is motivated by a number of reasons. First, cameras and IMUs provide exteroceptive and proprioceptive measurements, respectively, and thus are complementary to each other. Moreover, the IMU can provide high-frequency measurements (from hundreds to thousands of Hz), making it possible to obtain high-frequency pose estimates with low computational cost, simply by IMU integration. In addition, the camera can provide a very large amount of information (at the level of millions of pixels per image) to perform localization. Finally, the combination of the camera and IMU is inexpensive, light-weight, and consumes little power. These sensors can also be found in many mobile devices, and thus often do not require any additional hardware installation.

Visual-inertial localization methods can be placed in different categories, based on a number of criteria. Depending on the prior information of the features and the estimation objective, the localization algorithms can be divided into map-based methods [73,97], where the feature positions are known *a priori* and only the pose is estimated, simultaneous localization and mapping (SLAM) [41, 48, 82], where the feature positions and device pose are jointly estimated, and visual-inertial odometry (VIO) [64,74,103], where pose estimation

is performed without assuming or building a feature map. In this work, our goal is to track the motion of the platform in any environment, and we focus on the problem of VIO.

A different categorization of visual-inertial localization algorithms can be performed based on the way in which the visual measurements are used. One way is to process the images to extract features such as points [64], lines [53, 106] and planes [31]. Another is to employ the raw intensity measurements. In the latter case, depending on the type of image regions used in the localization, we can identify dense methods, where the entire image is used [77], semi-dense methods, where only regions with large gradient magnitude are used [20, 101], and patch-based methods, where regions around extracted point-features are used [26, 40, 94]. In all cases, the obtained camera measurements can be fused with the IMU measurements to obtain the pose estimates.

1.2 Key Contributions

While several algorithms for real-time VIO have been proposed, substantial research questions remain. In this work, we focus on improving the computational efficiency and accuracy of VIO. More specifically, the limited computational power of the CPU on many mobile platforms requires the estimator to efficiently use the computational resources. In some cases, a certain minor loss of accuracy may be acceptable, if it results in significant computational savings. Thus it is important to reduce the computational cost while maintaining high precision.

The reasons leading to loss of localization accuracy can come from different aspects. On the one hand, many existing localization algorithms either treat estimates from offline

calibration as known constants or simply ignore the non-ideal characteristics of the sensors, both of which can cause unmodelled errors and thus degrade accuracy. If a better modeling of the system is employed, an improved accuracy can be expected. On the other hand, we note that feature extraction from images unavoidably leads to information loss, as the images are being reduced to a small set of feature points. It is therefore possible to achieve accuracy improvement if we directly use the raw image intensities and avoid losing information. In what follows, we will propose algorithms to address each of the above problems.

1.2.1 Computationally efficient localization with the decoupled representation

The first contribution of the work is the novel parameterization of the trajectory of a moving platform, which facilitates the development of real-time pose-estimation methods [113]. In localization problems, the main challenge in reducing the computational cost is the high dimensionality of the estimation problem. In turn, a major reason for this is the large number of variables needed to represent the trajectory. A trajectory can be represented as the combination of the estimate and error. We notice that, in the existing trajectory representations, there is a one-to-one correspondence between the size of the estimated-state vector and error-state vector. We also note that an algorithm's computational cost mainly depends on the dimension of the error-state vector, due to the fact that the error-state vector determines the dimension of the estimate covariance or information matrix.

The key idea of the proposed approach is the *decoupling* of the parameterization of the trajectory *estimate* from the parameterization of the *error* in this estimate. Specifi-

cally, we represent the trajectory estimate as usual, via a set of pose states, each associated with a sensor reading (e.g., a laser scan or an image). The novelty of our approach lies in the representation of the estimation errors, for which we employ a continuous-time curve representation, namely B-splines. This decoupled formulation, which we term Decoupled Estimate-Error Parameterization (DEEP) offers two key advantages. First, the use of a high-dimensional pose-based representation of the trajectory allows us to represent arbitrarily complex trajectories. Second, the use of B-splines for error representation allows us to *control the computational complexity of an estimator*, by selecting the density of the knots of the B-spline. A low-dimensional spline representation of the errors can be used to reduce the computational cost, if desired. We empirically demonstrate that, in VIO, the DEEP formulation leads to substantial computational gains, while incurring only a small loss of estimation performance.

1.2.2 Improved estimation precision by high-fidelity sensor modeling and self-calibration

The second key contribution of this work is the high-fidelity sensor modeling and self-calibration in VIO, which improves estimation precision for systems equipped with low-cost inertial sensors and rolling-shutter cameras. When using visual and inertial measurements for state estimation, accurate knowledge of the sensor parameters is required. For this reason, the majority of existing approaches requires that prior offline calibration is performed in order to obtain values for some of the sensor characteristics (e.g., the intrinsic parameters of the camera), while others are often estimated online (e.g., the camera-to-IMU configuration), and yet others are assumed to be negligible (e.g., the misalignment of the

IMU axes). However, when treating the estimated values from offline calibration as known constants, one ignores the uncertainties of these parameters. Additionally, ignored non-ideal characteristics of the sensor can be significant, especially for low-cost senors. Both practices can thus result in unmodelled errors, which will inevitably degrade the estimation accuracy.

The key characteristic of the proposed method is that it performs *online self-calibration* of the camera and the IMU, using detailed models for both sensors and for their relative configuration. Specifically, the estimated parameters include the camera intrinsics (including lens distortion), the readout time of the rolling-shutter sensor, the IMU's biases, scale factors, axis misalignment, and g-sensitivity, the spatial configuration between the camera and IMU, as well as the time offset between the timestamps of the camera and IMU. An additional contribution of this work is a novel method for processing the measurements of the rolling-shutter camera, which employs an approximate representation of the estimation *errors*, instead of the state itself. We demonstrate, in both simulation tests and real-world experiments, that the proposed approach is able to accurately calibrate all the considered parameters in real time, and leads to significantly improved estimation precision compared to existing approaches.

1.2.3 Photometric patch-based VIO

Another key contribution of this work is a direct VIO algorithm, which utilizes image patches extracted around image features, and formulates measurement residuals in the image intensity space directly. While the direct methods avoid information loss and other drawbacks that exist in the feature-based approaches, they also suffer from illumination

changes caused by varying lighting conditions, viewing angle, etc. One key characteristic of the proposed method is the use of a *modified irradiance-consistency* constraint, where the true irradiance at each pixel is modeled as a random variable to be estimated and marginalized out. The formulation of the photometric residual explicitly accounts for the camera response function and lens vignetting (which can be calibrated in advance), as well as unknown illumination gains and biases, which are estimated on a per-feature or per-image basis. We present a detailed evaluation of our algorithm on 50 datasets with high-precision ground truth, which amount to approximately 1.5 hours of localization data. Through a direct comparison with a point-feature based method, we demonstrate that the use of photometric residuals results in increased pose estimation accuracy, with approximately 23% lower estimation errors, on average, in our data.

1.2.4 Semi-dense VIO

The results obtained in direct patch-based VIO show that, with proper modeling of the raw intensity measurements, improved estimation accuracy can be achieved in a direct formulation [114]. To further increase precision, we have extended direct VIO to a semi-dense framework, where all informative areas in images are used. Photometric triangulation is employed in this work, where the patch depth is estimated via the minimization of photometric residuals. Another contribution of the work is the novel noise model in the intensity measurement model, which accounts for noise during the image formation, as well as interpolation errors when acquiring intensities at non-integer pixel locations. Through both Monte-Carlo simulations and real-world experiments, we demonstrate that the proposed semi-dense VIO outperforms the direct VIO and the point-feature-based method, in

terms of estimation accuracy. We also justify that by using the photometric triangulation, a better accuracy can be achieved in the direct VIO compared to the one with the original, geometric triangulation.

1.3 Manuscript Organization

The remainder of this document is organized as follows. In chapter 2 we describe the DEEP formulation and apply it to an existing method in visual-inertial localization, namely the hybrid MSCKF/SLAM filter [62], along with online self-calibration. In chapter 3 we propose a localization algorithm that performs *online self-calibration* of the camera and the IMU, using high-fidelity models for both sensors and for their relative configuration. Chapter 4 analyzes the advantages and drawbacks of feature-based and direct approaches, and presents the photometric patch-based VIO algorithm. Chapter 5 further extends the direct VIO to a semi-dense framework, with photometric triangulation as well as a novel noise model. Finally, Chapter 6 concludes the dissertation.

Chapter 2

Efficient Pose Estimation with the DEEP formulation

2.1 Introduction

One of the key challenges in developing real-time VIO algorithms is the high dimensionality of the estimation problem at hand, which is caused by (i) the large number of variables (e.g., feature points) needed to model real-world environments, and (ii) the number of variables (i.e., platform poses) required to represent trajectories of long duration. In this chapter we focus on the latter issue, and propose a general method that can significantly reduce computational cost, while incurring only a small loss of accuracy.

We start by noting that the vast majority of practical localization algorithms are *linearization-based* methods, which typically rely on a variation of an extended Kalman filter (EKF), or iterative minimization. In these methods the computational cost is a function

of the dimension of the *error-state* vector. For instance, this dimension defines the size of the state-covariance matrix in an EKF, or the size of the Hessian in Newton-minimization methods. In current practice, the dimension of the estimator's error-state is directly determined by the number of poses in the trajectory, since one error-state vector (consisting, for instance, of the errors in position and orientation) is associated with each pose estimate. By contrast, in this work, we propose a new paradigm, where the number of poses included in the state vector and the dimension of the estimator's error-state are *decoupled*.

To describe the main idea of the method, we point out that any estimator that employs linearization relies on the computation of (i) measurement residuals, and (ii) linearized expressions showing the dependence of the residuals on the estimation errors. The first step involves the state *estimates*, while the second the state *errors*. The key insight here is that we may use *different representations* for each of these. Specifically, the state of a system at a certain time instant, t, is expressed as $\mathbf{x}(t) = \hat{\mathbf{x}}(t) + \tilde{\mathbf{x}}(t)$, where $\hat{\mathbf{x}}(t)$ is the state estimate, and $\tilde{\mathbf{x}}(t)$ is the error in this estimate. In order to represent the estimates, we use the "traditional" representation, where one pose vector (e.g., consisting of position and orientation) is maintained for each time instant at which measurements are available. This representation is preferable, as it makes no assumptions on the form of the trajectory. On the other hand, for the error-state $\tilde{\mathbf{x}}(t)$, we employ a continuous-time representation using temporal B-spline functions. Since the estimation errors are expressed as a function of the B-spline control points, this in turn means that we can express all the estimator Jacobians as a function of these parameters.

It has been demonstrated that, due to the nature of the estimation errors, a *low-dimensional* B-spline representation suffices in order to describe the errors well [113]. Therefore, the estimator equations can be expressed in terms of a small number of B-spline control points (which now constitute the error-state vector of the estimator), leading to a reduction in computational cost. We term the proposed approach, in which a different representation is used for the trajectory estimates and their errors, Decoupled Estimate-Error Parameterization (DEEP).

The proposed formulation is a general one and can be employed in several classes of estimation problems. We here formulate a hybrid MSCKF/SLAM filter in the proposed DEEP framework [62], to solve the problem of 3D localization with *online* visual-inertial calibration by using image measurements from a rolling-shutter camera.

The ability of doing online sensor calibration is crucial for a high-precision estimator. Many prior works focus on offline calibration [34, 72, 110]. However, the main drawback of offline calibration is that, even if a high-quality estimate of a given parameter is obtained, this is still not the *true* parameter value. Thus, treating the estimate from offline calibration as the true value leads to unmodeled errors, and degrades estimation accuracy. In contrast, we concurrently localize and estimate the following quantities *online*:

- the IMU biases
- the camera-to-IMU spatial configuration, including the rotation and translation
- the camera intrinsics, including the lens distortion
- the image readout time of the rolling-shutter camera

- the time offset between timestamps of the camera and the IMU

Through both Monte-Carlo simulations and real-world experiments, we show that the DEEP formulation yields substantial gains in computational efficiency, while only incurring a small loss of accuracy. In our simulation tests, for example, the proposed approach is shown to reduce the computational cost (measured in terms of the runtime per image) by 84%, while increasing the estimation errors by a mere 1.7% compared with a high-order Taylor-series approximation for modeling the camera motion; or to reduce the cost by 60%, while increasing the estimation errors by less than 1% compared with a low-order Taylor-series approximation. More importantly, we demonstrate that the approach leads to a graceful degradation of performance, as the dimensionality of the error representation is reduced.

2.2 Related Work

To the best of our knowledge, the topic of joint calibration of the visual-inertial sensors by using a continuous-time representation of the *errors* has not been addressed in the past. In what follows, we discuss the relevant works, in two categories:

a) Computational complexity reduction: At a high level, most computation-reduction methods for EKF-based algorithms propose either exact or approximate reformulations of the estimator equations to reduce the computational cost of updating the filter's state-covariance matrix (see, e.g., [30, 42, 75, 81] and references therein). On the other hand, to reduce the computational cost of graph-based methods, key approaches have included exploiting the sparsity of the graph, improving the computational characteristics of the

minimization algorithm, and focusing computation on only the relevant parts of the graph (e.g., [15, 43, 44, 56]). By contrast, we here focus on reducing dimensionality via changing the representation of the error-state, which is conceptually different. The DEEP formulation can, in principle, be employed with *any* linearization-based method which maintains estimates of a platform's trajectory, including the ones mentioned above.

Our approach is more closely related to methods which seek to reduce the number of poses included in the state vector in localization. This is typically accomplished either by simply using measurements less frequently, or by intelligently selecting only a subset of *keyframes* to include in the estimator [50, 76], or by pruning an already existing pose graph [55]. Our proposed approach differs in a key aspect: we do *not* aim at reducing the number of poses included in the estimated state vector. Instead, we decouple the representation of the trajectory (which is still represented as a set of poses) from the representation of the trajectory errors (which are represented by B-splines), and aim at *reducing the dimensionality of the error state only*. In our approach, the measurements recorded from all poses are processed, using a reduced-dimension representation for the pose errors. As a result, we avoid the loss of information that occurs when a number of poses and their associated measurements are discarded from the estimator.

The representation of the error states in localization has been studied in a number of previous works, which focus on addressing the fact that (most) orientation representations involve differentiable manifolds [25, 33, 38, 90, 91]. However, in all these works, there exists a one-to-one correspondence between the number of state estimates and error states, and thus these approaches are not closely related to our work. The same holds for works where new

state parameterizations are proposed (see, e.g. [12, 70]), resulting in a subsequent change in the representation of the state errors.

The approaches most closely related to ours are those that employ temporal basis functions to represent the trajectory in continuous time. Earlier works using spline-based representations of the trajectory can be found in [4, 6, 7]. The methodology was formalized in [27], and has been employed for pose estimation and calibration of rolling-shutter cameras in [79], and for visual-inertial SLAM in [68]. This continuous-time representation of the trajectory offers the advantage that, by increasing the number of basis functions (and thus the computational cost), one can model arbitrarily complex trajectories. A decomposition of the trajectory and the correction is employed in [115], in which the trajectory and the correction all share continuous-time representations but with different number of control points. The DEEP formulation we use in this paper was first proposed in [113], in which the representations (and thus the sizes) of the trajectory and the estimation error are decoupled. This allows us to model arbitrarily complex trajectories at a lower computational cost (see Section 2.7.2), and unlike the above works where continuous-time representations were used in a batch optimization method, [113] was the first one to employ the continuous-time representation in an EKF-based filter.

b) Visual-inertial calibration: In many previous works, camera intrinsic parameters are assumed to be known constants (e.g. via an offline calibration, see [34, 110] and the references therein), but these are all visual-based approaches. The calibration of the spacial configuration between the IMU and camera has been studied in a number of works

(see [41,48,64,103] and references within), whereas the temporal relationship between them is less explored [49,59].

The use of a rolling-shutter camera for localization requires additional parameter calibration: the image readout time (i.e. the time needed to capture all rows in one image). The readout time is usually obtained by offline calibration, which requires prior knowledge of the environment (e.g. a known calibration pattern in [79], a LED flashing at a known frequency in [72]). Recently, more works focus on calibrating the readout time by fusing the camera and IMU measurements [32,65,66,104], or only the camera and gyroscope measurements [39]. The advantage of these works is that no information about the environment is required.

The approach most closely related to our approach on calibration is that of [80], where a continuous-time representation of the trajectory is used in 3D localization as well as the visual-inertial calibration. However, the time offset between the timestamps of the IMU and camera is not considered in this work, and the representation of the trajectory differs from our formulation.

2.3 Continuous-Time Error-State Representation

In this section, we present the main idea of the DEEP formulation, which relies on a low-dimensional, continuous-time representation of the trajectory errors. As discussed in Section 2.1, our choice is to employ a B-spline representation of the errors. Similarly to prior work [27], this is motivated by the need to have temporal basis functions with local

support and simple analytical derivatives. In what follows, we briefly present the B-spline representation of a function, and then describe the way in which it is employed in our work.

2.3.1 B-Spline function representation

A B-spline function of degree k is a piecewise-polynomial function of degree k, defined over an interval $[t_0,\ t_n]$. The points $t_i, i = 0, \ldots n$, are termed the *knots* of the B-spline, and in our work we assume that the knots are uniformly distributed (i.e., we employ a uniform B-spline). We employ quadratic and cubic B-splines. Over the time interval $[t_i, t_{i+1}]$, a quadratic B-spline function is given by [87]:

$$r_{2,i}(u) = \underbrace{\frac{1}{2}\begin{bmatrix} u^2 & u & 1 \end{bmatrix}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{B}_2(u)}\begin{bmatrix} c_i \\ c_{i+1} \\ c_{i+2} \end{bmatrix} \tag{2.1}$$

where $u = (t - t_i)/\delta t$, $\delta t = \frac{t_n - t_0}{n}$. In the above equation, c_i, c_{i+1} and c_{i+2} are the parameters, termed *control points*, which determine the form of the polynomial function. Similarly, a cubic B-spline function over the interval $[t_i, t_{i+1}]$ is defined as:

$$r_{3,i}(u) = \underbrace{\frac{1}{2}\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}}_{\mathbf{B}_3(u)}\begin{bmatrix} c_i \\ c_{i+1} \\ c_{i+2} \\ c_{i+3} \end{bmatrix} \tag{2.2}$$

The above equations define the B-spline functions in each of the intervals between knots. The value of the function at any time $t \in [t_0, t_n)$ is given by:

$$r(t) = \begin{bmatrix} \mathbf{0} & \mathbf{B}_k(u) & \mathbf{0} \end{bmatrix} \mathbf{c} \tag{2.3}$$

$$= \bar{\mathbf{B}}_k(t)\mathbf{c} \tag{2.4}$$

where k denotes the chosen degree of the B-spline, and $\mathbf{c}$ is the vector containing all the control points c_i, $i = 0, \ldots, n+k-1$. Note that the relationship between the function value, $r(t)$, and $\mathbf{c}$ is linear, which will be important for our formulation.

2.3.2 Key idea

B-splines are useful tools for representing (or approximating) a smooth function, a fact that motivated their use for trajectory modeling in [27] and derivative approaches. In these formulations, the dimension of the state vector, and therefore the computational cost of the estimator, depends on the spacing of the knots. When the knots are spaced closely in time, we are able to model complex trajectories. However, using a large number of knots increases the dimension of the state vector and the computational requirements, which is a key drawback. Moreover, since the required dimension of the knot vector depends on the trajectory itself, it may be hard to make this choice in advance (this is also discussed in [79]).

To motivate the DEEP formulation proposed here, in Fig. 2.1 we present the trajectory (x-y-z position) of a platform during a one-second long window from one of our simulations, which involves localization using visual and inertial measurements. Specifically, in these plots the blue solid lines represent the true trajectory; the red dash-dotted lines

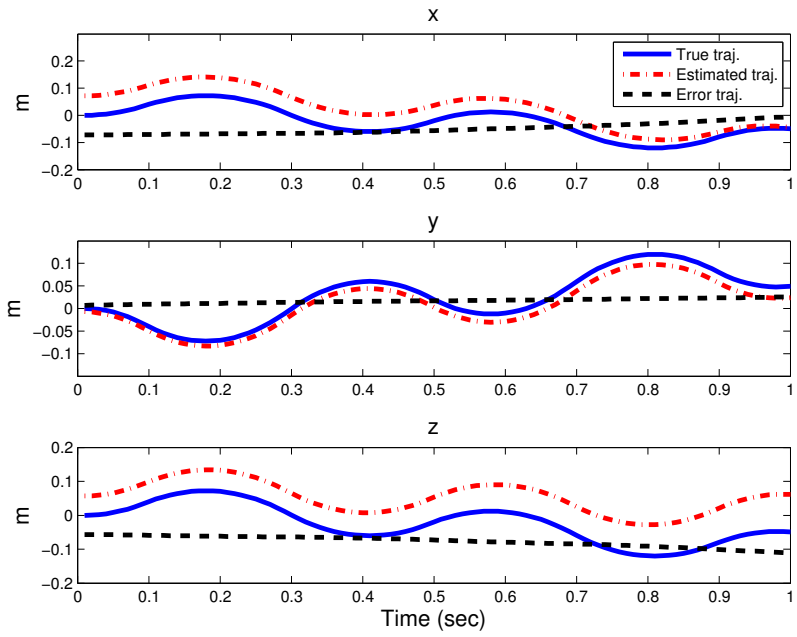

Figure 2.1: Example trajectory and estimation errors during a one-second long interval of visual-inertial localization.

represent the trajectory estimate computed via inertial propagation in this time interval, starting from an initial inaccurate estimate; and the black dashed lines represent the errors in the estimates (i.e., the difference between the true and estimated position). The key point to notice in this plot is that, while the actual trajectory is quite "complex" in this time interval, the functions describing the estimation errors are much "smoother". In turn, this means that in order to obtain an accurate B-spline approximation of the actual trajectory, a *significantly larger knot vector would be required*, compared to that needed for a B-spline approximation to the trajectory errors. Our approach takes advantage of this observation.

Specifically, consider a platform equipped with proprioceptive sensors (e.g., an inertial measurement unit (IMU)) as well as exteroceptive ones (e.g., a camera). Let us

assume that during a time interval $[t_0, t_n]$, M exteroceptive measurements are recorded (e.g., M images). In the DEEP formulation, the estimator maintains an estimate for the platform states (e.g., poses) corresponding to these M time instants, $\hat{\mathbf{x}}_i$, $i = 1, \dots M$. An initial estimate for each state is computed via propagation (e.g., integration of the inertial measurements) from the latest available pose, and updated as more measurement information is processed. The novelty of our approach lies in the representation of the errors. In particular, we model the state error, $\tilde{\mathbf{x}}(t)$, $t \in [t_0, t_n]$ via a B-spline representation. As a result, the error in the estimate of the i-th pose, $\tilde{\mathbf{x}}_i$, which is required in the derivation of the linearized residual equations in the estimator, can be written as a linear function of the control-point vector (see (2.3)). Since all linearized residuals can be written in terms of this vector, it constitutes the "error-state vector" of the estimator.

If the number of knots needed to accurately describe the trajectory errors is smaller than M, then the dimension of the error-state vector in the DEEP formulation will be smaller than that of the "traditional" approach in which M pose errors are explicitly represented. As shown in the following sections, at least for the problem of visual-inertial localization, we can use a number of knots that is significantly smaller than M (e.g., five to ten times smaller), without incurring significant loss of estimation accuracy. The justification for this lies in the fact that, as shown in Fig. 2.1, the estimation errors tend to be "smooth" over short periods of time. Moreover, we note that the "complexity" of the estimation error is, to a large extent, independent from the complexity of the trajectory itself. This is useful, since it allows us to choose the knot density without having prior knowledge of the trajectory.

2.4 Camera Model

We here present the measurement model of the rolling-shutter camera and introduce all the calibration parameters needed. Let us consider that an image is received with timestamp t, which corresponds to the time instant the middle image row is read. Due to hardware limitations, the time instant when we receive the sensor measurement will be different from the time when it is actually measured, resulting in time delays for each sensor. Due to different time delays for each sensor, the image timestamp is affected by a time offset, t_d, relative to the IMU timestamps [59]. Therefore, the middle image row was actually captured at time $t + t_d$, and an image row that is n rows away from the middle was captured at

$$t_n = t + t_d + \frac{nt_r}{U}$$

where t_r is the image readout time, and U is the total number of image rows (namely, the image height). Note that n is a signed quantity: it is positive for rows below the middle, and negative for rows above it.

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:

$$\mathbf{z} = \mathbf{h}({}^C\mathbf{p}_f(t_n)) + \mathbf{n} \tag{2.5}$$

where ${}^C\mathbf{p}_f(t_n)$ is the position of the feature with respect to the camera frame {C} at time t_n, $\mathbf{n}$ is the measurement noise vector, modelled as zero-mean white Gaussian noise with covariance matrix $\sigma^2\mathbf{I}_2$, and $\mathbf{h}(\cdot)$ is the camera's projection function, which is a modeled

as the standard perspective camera with radial and tangential distortions [37]. Denoting ${}^{C}\mathbf{p}_f(t_n) = [{}^{C}x_f \;\; {}^{C}y_f \;\; {}^{C}z_f]^T$, the image projection model is given by:

$$\mathbf{h}\left({}^{C}\mathbf{p}_f\right) = \mathbf{p}_c + \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix} \begin{bmatrix} u_d \\ v_d \end{bmatrix} \tag{2.6}$$

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

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = d \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2t_1uv + t_2(u^2+v^2+2uv) \\ t_1(u^2+v^2+2uv) + 2t_2uv \end{bmatrix} \tag{2.7}$$

$$d = 1 + k_1(u^2+v^2) + k_2(u^2+v^2)^2 + k_3(u^2+v^2)^3$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^{C}z_f} \begin{bmatrix} {}^{C}x_f \\ {}^{C}y_f \end{bmatrix} \tag{2.8}$$

In the above, k_i, $i = 1, 2, 3$ are the radial distortion coefficients, while t_1, t_2 are the tangential distortion coefficients. Moreover, the position vector ${}^{C}\mathbf{p}_f(t_n)$ can be written as:

$${}^{C}\mathbf{p}_f(t_n) = {}^{C}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}(t_n)\left({}^{G}\mathbf{p}_f - {}^{G}\mathbf{p}_I(t_n)\right) + {}^{C}\mathbf{p}_I \tag{2.9}$$

where ${}^{C}_{I}\mathbf{R}$ is the rotation matrix from frame $\{I\}$ to $\{C\}$, which can be represented by an unit quaternion ${}^{C}_{I}\mathbf{q}$; ${}^{C}\mathbf{p}_I$ is the position of the origin of $\{I\}$ in the camera frame. Since all of these parameters are unknown , they must also be estimated in the filter. We can now define the 15×1 vector containing the calibration parameters estimated in the EKF:

$$\mathbf{x_c} = \begin{bmatrix} {}^{C}_{I}\mathbf{q}^T & {}^{C}\mathbf{p}_I^T & \mathbf{p}_c^T & a_u & a_v & k_1 & k_2 & k_3 & t_1 & t_2 & t_r & t_d \end{bmatrix}^T \tag{2.10}$$

The estimation of these parameters proceeds as normal in an EKF, by computing the Jacobians of the camera measurement model with respect to these parameters, and updating their estimates during EKF updates.

2.4.1 Rolling-shutter model: prior work

Before we show our approach to process feature measurements, it is interesting to explore the existing methods to deal with the rolling-shutter effect. In a rolling-shutter camera, image measurements are captured row by row, at slightly different time instants, and thus, feature measurements at different rows in the *same* image depend on poses at *different* time instants (see (2.9)). Since it would be computationally impractical to include all these poses into the state vector, existing approaches usually employ assumptions on the form of the trajectory during the image readout time (e.g. constant velocity model [35, 58], linear interpolation model [32, 104], or higher-order parametric representations [36, 79]). However, these methods either result in loss of modelling fidelity (when using low-dimensional models), or high computational cost (when using high-dimensional ones).

A better approach was employed in [66,107], where no assumption about the form of the trajectory is applied, and only an approximation on the *error* during the image readout time is used. Specifically, the error during the readout time (at time $\hat{t}_n$) is approximated as a function of the error-states when the center row of the image was captured (at time $t + \hat{t}_d$). More specifically, by expanding the Taylor-series of the pose error at $\hat{t}_n$, the error can be written as:

$$
\begin{aligned}
{}^{G}\tilde{\mathbf{p}}(\hat{t}_n) &= \sum_{i=0}^{\infty} \frac{(n\hat{t}_r)^i}{U^i i!} {}^{G}\tilde{\mathbf{p}}^{(i)}(t+\hat{t}_d) \\
&= {}^{G}\tilde{\mathbf{p}}(t+\hat{t}_d) + \frac{n\hat{t}_r}{U} {}^{G}\tilde{\mathbf{v}}(t+\hat{t}_d) + \frac{(n\hat{t}_r)^2}{2U^2} {}^{G}\tilde{\mathbf{a}}(t+\hat{t}_d) + \dots \\
{}^{G}\tilde{\boldsymbol{\theta}}(\hat{t}_n) &= \sum_{i=0}^{\infty} \frac{(n\hat{t}_r)^i}{U^i i!} {}^{G}\tilde{\boldsymbol{\theta}}^{(i)}(t+\hat{t}_d) \\
&= {}^{G}\tilde{\boldsymbol{\theta}}(t+\hat{t}_d) + \frac{n\hat{t}_r}{U} \tilde{\boldsymbol{\omega}}(t+\hat{t}_d) + \dots
\end{aligned}
$$

These expressions are exact if all terms are kept, but to achieve computational tractability, only a finite number of terms are kept in the estimator:

$$^{G}\tilde{\mathbf{p}}_I(\hat{t}_n) \simeq \sum_{i=0}^{I_{\mathbf{p}}} \frac{(n\hat{t}_r)^i}{U^i i!} {}^{G}\tilde{\mathbf{p}}^{(i)}(t+\hat{t}_d) \tag{2.11}$$

$$^{G}\tilde{\boldsymbol{\theta}}(\hat{t}_n) \simeq \sum_{i=0}^{I_{\boldsymbol{\theta}}} \frac{(n\hat{t}_r)^i}{U^i i!} {}^{G}\tilde{\boldsymbol{\theta}}^{(i)}(t+\hat{t}_d) \tag{2.12}$$

where $I_{\mathbf{p}}, I_{\boldsymbol{\theta}}$ are the truncation orders kept for the position and orientation errors, respectively. Since these equations express the errors at $\hat{t}_n$ as function of the errors at $t + \hat{t}_d$, which are already in the error-state vector, this approximation can be employed into (2.14) and used in the EKF update. Because this error-approximation approach has been shown to outperform prior methods in processing rolling-shutter measurements, we will use it as a benchmark to test our approach.

2.4.2 Processing of rolling-shutter measurements in the DEEP formulation

Processing the rolling-shutter measurements in the DEEP formulation is similar to processing global-shutter measurements. Specifically, recall that in the DEEP approach there is *no* assumption imposed on the form of the trajectory. The B-spline representation of the estimation errors can naturally express the errors during the image readout time, so no additional assumptions are needed, compared with the global-shutter case. Moreover, since the B-spline representation naturally includes the information of the velocity, acceleration and angular velocity [1], there is no need to include these higher-order terms from Taylor-series into the error-state vector as in [66]. As shown in Section 2.7, the DEEP formulation

[1]This is valid for cubic or higher-order B-spline representations of position errors, and quadratic or higher-order B-spline representations of orientation errors.

results in accuracy that is similar to that of [66], at a lower computational cost, or higher accuracy for similar cost.

Now we give the detailed description of processing the rolling-shutter measurements in DEEP formulation. Note that in order to use a measurement in an EKF update, the processing of the measurement requires: (i) the residual of the measurement; (ii) a linear expression that relates the measurement residual to the error states in the error-state vector. The residual corresponding to the feature measurement model (2.5) is defined as:

$$\mathbf{r} = \mathbf{z} - \mathbf{h}\left({}^{C}_{I}\hat{\mathbf{R}}\,{}^{I}_{G}\hat{\mathbf{R}}(\hat{t}_n)\left({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_I(\hat{t}_n)\right) + {}^{C}\hat{\mathbf{p}}_I\right) \tag{2.13}$$

where $\hat{t}_n = t + \hat{t}_d + \frac{n\hat{t}_r}{U}$. Denote $\hat{\boldsymbol{\pi}}(\hat{t}_n) = [{}^{I}_{G}\hat{\bar{\mathbf{q}}}(\hat{t}_n)^T\ {}^{G}\hat{\mathbf{p}}(\hat{t}_n)^T]^T$, which is the estimate of the IMU pose at the time instant $\hat{t}_n$. In order to evaluate the residual, we compute $\hat{\boldsymbol{\pi}}(\hat{t}_n)$ by integrating the IMU measurements at the time interval $[t+\hat{t}_d, \hat{t}_n]$, starting from $\hat{\boldsymbol{\pi}}(t+\hat{t}_d)$ (note that backward integration may be necessary). The advantage of doing so is that, by employing direct integration of the IMU measurements to compute $\hat{\boldsymbol{\pi}}(\hat{t}_n)$, arbitrary motions can be described (as long as they are within the bandwidth of the IMU). Once $\hat{\boldsymbol{\pi}}(\hat{t}_n)$ is computed, it is included into the *estimated* state vector (see (2.20)).

To derive the linear equation relating the residual to the error states, we first linearize the camera observation model in (2.13), to obtain:

$$\mathbf{r} \simeq \mathbf{H}_{\mathbf{r}\boldsymbol{\theta}}\,{}^{G}\tilde{\boldsymbol{\theta}}(\hat{t}_n) + \mathbf{H}_{\mathbf{rp}}\,{}^{G}\tilde{\mathbf{p}}(\hat{t}_n) + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}}\,{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{2.14}$$

where $\mathbf{H}_{\mathbf{r}\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{rp}}$ are the Jacobians of the measurement function with respect to the orientation and position errors at time $\hat{t}_n$, respectively, $\mathbf{H}_{\mathbf{c}}$ is the Jacobian with respect to $\mathbf{x}_{\mathbf{c}}$, and $\mathbf{H}_{\mathbf{f}}$ is the Jacobian with respect to the feature position. Assuming the update

takes place at time-step $k+N$, we can write the pose errors ${}^{G}\tilde{\boldsymbol{\theta}}(\hat{t}_n)$ and ${}^{G}\tilde{\mathbf{p}}(\hat{t}_n)$ as a *linear* function of the error-state:

$$
\begin{aligned}
{}^{G}\tilde{\boldsymbol{\theta}}(\hat{t}_n) &= \mathbf{B}_{\boldsymbol{\theta}}(\hat{t}_n)\mathbf{x}_{\boldsymbol{\theta}_{k+N}} \\
{}^{G}\tilde{\mathbf{p}}(\hat{t}_n) &= \mathbf{B}_{\mathbf{p}}(\hat{t}_n)\mathbf{x}_{\mathbf{p}_{k+N}}
\end{aligned}
$$

Thus, (2.14) becomes:

$$
\mathbf{r} \simeq \mathbf{H}_{\boldsymbol{\theta}}(\hat{t}_n)\mathbf{x}_{\boldsymbol{\theta}_{k+N}} + \mathbf{H}_{\mathbf{p}}(\hat{t}_n)\mathbf{x}_{\mathbf{p}_{k+N}} + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{2.15}
$$

where $\mathbf{H}_{\boldsymbol{\theta}}(\hat{t}_n) = \mathbf{H}_{\mathbf{r}\boldsymbol{\theta}}\mathbf{B}_{\boldsymbol{\theta}}(\hat{t}_n)$ and $\mathbf{H}_{\mathbf{p}}(\hat{t}_n) = \mathbf{H}_{\mathbf{rp}}\mathbf{B}_{\mathbf{p}}(\hat{t}_n)$ are the Jacobians of the measurement function with respect to orientation and position control points, respectively. This expression can be used for the EKF update, as detailed in Section 2.5.4.

2.5 Visual-inertial Localization in the DEEP Formulation

We now describe a visual-inertial localization algorithm that employs the DEEP approach described in Section 2.3. The algorithm we describe here is an adaptation of the "hybrid" EKF algorithm which was originally proposed in [62]. It integrates the sliding-window approach of the MSCKF 2.0 algorithm [64, 74] and a feature-based formulation (EKF-SLAM), and thus can exploit the computational advantages of both. The hybrid filter has been shown in prior work to achieve state-of-the-art accuracy in VIO, and thus can be used as a benchmark for examining the performance of the DEEP formulation.

In the original hybrid estimator, the state vector consists of a sliding window of M camera poses, which correspond to the time instants the last M images were recorded, as well as the features. The key characteristic of the approach is that, for features whose

feature track length are less than M, their measurements are used to apply constraints on the poses of the sliding window, while the ones with longer feature track length are initialized into the state vector to perform the standard EKF-SLAM update. The hybrid DEEP-EKF estimator we describe here follows the same approach.

2.5.1 IMU state definition

Consider a platform equipped with an IMU and a monocular camera, moving in an area populated with naturally-occurring point features, whose coordinates are not known a priori. Our goal is to estimate the position and orientation of the platform with respect to a global coordinate frame, $\{G\}$, using the inertial measurements and observations of these features. To derive the estimator's equations, we affix a coordinate frame $\{I\}$ to the IMU, and a coordinate frame $\{C\}$ to the camera. The camera intrinsics and the spatial and temporal relationships between frame $\{I\}$ and $\{C\}$ are unknown, and thus need to be calibrated online.

The IMU state at time-step k is described by the vector:

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_k^T & {}^{G}\mathbf{v}_k^T & \mathbf{b}_{\mathbf{g}_k}^T & \mathbf{b}_{\mathbf{a}_k}^T \end{bmatrix}^T \tag{2.16}$$

where[2] ${}^{I_k}_{G}\bar{\mathbf{q}}$ is the unit quaternion [98] representing the rotation from the global frame $\{G\}$ to the IMU frame $\{I\}$ at time-step k, ${}^{G}\mathbf{p}_k$ and ${}^{G}\mathbf{v}_k$ 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, respectively, which are modeled as Gaussian random-walk processes.

[2]Notation: The preceding superscript for vectors (e.g., X in ${}^{X}\mathbf{a}$) denotes the frame of reference with respect to which quantities are expressed. ${}^{X}_{Y}\mathbf{R}$ is the rotation matrix rotating vectors from $\{Y\}$ to $\{X\}$, and ${}^{X}_{Y}\bar{\mathbf{q}}$ is the corresponding unit quaternion. ${}^{X}\mathbf{p}_Y$ is the origin of frame $\{Y\}$ with respect to $\{X\}$. $\mathbf{0}$ and $\mathbf{I}$ are the zero and identity matrices respectively, while $\hat{a}$ and $\tilde{a}$ represent the estimate, and error of the estimate, of a variable a, respectively.

The error in the estimate of the IMU state (2.16) is defined as [64]:

$$\tilde{\mathbf{x}}_{I_k} = \begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_k^T & {}^G\tilde{\mathbf{p}}_k^T & {}^G\tilde{\mathbf{v}}_k^T & \tilde{\mathbf{b}}_{\mathbf{g}_k}^T & \tilde{\mathbf{b}}_{\mathbf{a}_k}^T \end{bmatrix}^T \tag{2.17}$$

where the standard additive error definition is used for the position, velocity and biases (i.e., if $\hat{\mathbf{y}}$ is the estimate of a variable $\mathbf{y}$, the estimation error is defined as $\tilde{\mathbf{y}} = \mathbf{y} - \hat{\mathbf{y}}$), while for the orientation errors we use a minimal 3-dimensional representation, defined by the equation:

$${}^{I_k}_{G}\bar{\mathbf{q}} \approx {}^{I_k}_{G}\hat{\bar{\mathbf{q}}} \otimes \begin{bmatrix} \frac{1}{2}{}^G\tilde{\boldsymbol{\theta}}_k \\ 1 \end{bmatrix} \tag{2.18}$$

where $\otimes$ denotes quaternion multiplication.

2.5.2 State vector of the hybrid DEEP-EKF

The *estimated* state vector of the hybrid DEEP-EKF at time-step k is defined as:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\mathbf{x}}_{I_k}^T & \hat{\mathbf{x}}_c^T & \hat{\boldsymbol{\pi}}_{k-1}^T & \hat{\boldsymbol{\pi}}_{k-2}^T & \cdots & \hat{\boldsymbol{\pi}}_{k-M}^T & \hat{\mathbf{f}}_1^T \cdots & \hat{\mathbf{f}}_s^T \end{bmatrix}^T \tag{2.19}$$

where $\hat{\mathbf{x}}_c$ is the vector of parameters we seek to calibrate, defined in (2.10); $\mathbf{f}_j$, for $j = 1, \ldots, s$ are the estimates of the current visible features, in inverse-depth parameterization; and $\hat{\boldsymbol{\pi}}_i$, for $i = k - M, \ldots, k - 1$, are the estimates of the IMU poses at the time instants the last M images were recorded. Note that when using a global-shutter camera, all image measurements in one image are taken at the *same* time, and thus there is only one pose corresponding to each image, namely $\hat{\boldsymbol{\pi}}_i = [{}^{I_i}_{G}\hat{\bar{\mathbf{q}}}^T \quad {}^G\hat{\mathbf{p}}_i^T]^T$. However, when using the rolling-shutter camera, measurements on different rows in the same image are taken at *different* time instants, thus $\hat{\boldsymbol{\pi}}_i$ contains poses corresponding to all rows in the i-th image, namely,

$$\hat{\boldsymbol{\pi}}_i = \begin{bmatrix} \hat{\boldsymbol{\pi}}_i(\hat{t}_1)^T & \ldots & \hat{\boldsymbol{\pi}}_i(\hat{t}_U)^T \end{bmatrix}^T \tag{2.20}$$

where $\hat{\boldsymbol{\pi}}_i(\hat{t}_n)$, $i = 1 \dots U$, is the IMU pose estimate corresponding to the n-th row in the i-th image.

The above estimated state vector is identical to that of the original hybrid EKF. The difference lies in the representation of the errors in the M sets of poses included in the estimated state vector. Specifically, instead of maintaining an individual error state for each of the M poses, in the DEEP formulation we model the position and orientation errors in the time interval when these M sets of poses were recorded by B-spline functions. The error-state of the hybrid DEEP-EKF therefore contains the control points of these B-splines. Specifically, the error-state vector at time-step k is defined as:

$$\tilde{\mathbf{x}}_k = [\tilde{\mathbf{b}}_{\mathbf{g}_k}^T \ \tilde{\mathbf{b}}_{\mathbf{a}_k}^T \ \tilde{\mathbf{x}}_c^T \underbrace{\mathbf{c}_{\mathbf{p}_k}^T \ \mathbf{c}_{\mathbf{p}_{k-N}}^T \cdots \mathbf{c}_{\mathbf{p}_{k-\ell N}}^T}_{\mathbf{x}_{\mathbf{p}_k}^T} \underbrace{\mathbf{c}_{\boldsymbol{\theta}_k}^T \ \mathbf{c}_{\boldsymbol{\theta}_{k-N}}^T \cdots \mathbf{c}_{\boldsymbol{\theta}_{k-(\ell-1)N}}^T}_{\mathbf{x}_{\boldsymbol{\theta}_k}^T} \tilde{\mathbf{f}}_1^T \cdots \tilde{\mathbf{f}}_s^T]^T \tag{2.21}$$

where $\mathbf{c}_{\mathbf{p}_j}$, $j = k - N\ell, k - N\ell + N, .., k$ and $\mathbf{c}_{\boldsymbol{\theta}_j}$, $j = k - (\ell - 1)N, k - (\ell - 2)N, .., k$ are 3×1 control-point vectors used to represent the position and orientation errors, respectively. To simplify the presentation, we here assume that a new knot is introduced every N camera images (note however, that having the knot interval be an integer multiple of the image period is not a strict requirement). In effect, N here is a "knot-spacing factor", which determines the number of knots needed to cover the time interval in which the latest M images were recorded. Increasing N results in smaller size of the error-state vector (and thus lower computational cost), and vice versa. Note that in (2.21) the number of position control points is larger than the number of orientation control points by one. This is due to the fact that in our implementation we employ a quadratic B-spline to represent the orientation errors, and a cubic B-spline to represent the position errors. This choice was dictated by the fact that the IMU accelerometer provides measurements of the second-order derivative

of the position, while the gyroscope provides measurements of the first-order derivative of the orientation.

With this representation, the errors of each individual pose in the estimated state vector in (2.19) are expressed as:

$$^{G}\tilde{\boldsymbol{\theta}}_j = \mathbf{B}_{\boldsymbol{\theta}}(t_j)\mathbf{x}_{\boldsymbol{\theta}_k} \tag{2.22}$$

$$^{G}\tilde{\mathbf{p}}_j = \mathbf{B}_{\mathbf{p}}(t_j)\mathbf{x}_{\mathbf{p}_k} \tag{2.23}$$

where $\mathbf{B}_{\boldsymbol{\theta}}(t_j) = \text{kron}(\bar{\mathbf{B}}_2(t_j), \mathbf{I}_3)$ and $\mathbf{B}_{\mathbf{p}}(t_j) = \text{kron}(\bar{\mathbf{B}}_3(t_j), \mathbf{I}_3)$, with kron denoting the Kronecker matrix product, and $\mathbf{I}_3$ the 3×3 identity matrix. Moreover, we note that the errors in the velocity can also be written as a linear function of the control points:

$$^{G}\tilde{\mathbf{v}}_j = \mathbf{B}^{(1)}_{\mathbf{p}}(t_j)\mathbf{x}_{\mathbf{p}_k} \tag{2.24}$$

where $\mathbf{B}^{(1)}_{\mathbf{p}}(t_j)$ is the time-derivative of $\mathbf{B}_{\mathbf{p}}(t_j)$.

2.5.3 State augmentation

The hybrid DEEP-EKF filter proceeds as follows: let us consider that at time-step k (i.e., time t_k) a filter update is performed (see Section 2.5.4). From that point on, the estimator uses the IMU measurements for propagating the IMU state estimate $\hat{\mathbf{x}}_I$, using the integration equations described in [64]. Every time a new image is recorded, a new pose estimate $\hat{\boldsymbol{\pi}}_j$ is included in the estimated state vector (2.19), but the error-state, and corresponding covariance matrix, remains unchanged. A new set of control points is

introduced in the error-state vector (2.21) at time t_{k+N}, i.e, once N images have been recorded. At this time, the error-state is augmented by including the vector:

$$\tilde{\mathbf{x}}_{new} = \begin{bmatrix} \tilde{\mathbf{b}}_{\mathbf{g}_{k+N}}^T & \tilde{\mathbf{b}}_{\mathbf{a}_{k+N}}^T & \mathbf{c}_{\mathbf{p}_{k+N}}^T & \mathbf{c}_{\boldsymbol{\theta}_{k+N}}^T \end{bmatrix}^T \tag{2.25}$$

which consists of the errors of the IMU biases at the current time, as well as the control points needed for representing the position and orientation errors in the time interval $[t_k, t_{k+N}]$.

To complete the augmentation, we must also augment the covariance matrix of the EKF. To this end, we must compute the covariance matrix of $\tilde{\mathbf{x}}_{new}$, as well as its cross-correlation with the other error-states. We begin with the expression relating the IMU-pose errors at time t_{k+N} and t_k:

$$\tilde{\mathbf{x}}_{I_{k+N}} = \boldsymbol{\Phi}_I(t_{k+N}, t_k)\tilde{\mathbf{x}}_{I_k} + \mathbf{w}_k \tag{2.26}$$

where $\mathbf{w}_k$ is the process-noise error, which is zero-mean Gaussian with covariance matrix $\mathbf{Q}_k$. The computation of $\boldsymbol{\Phi}_I(t_{k+N}, t_k)$ and $\mathbf{Q}_k$ is described in [64]. To obtain an expression relating $\tilde{\mathbf{x}}_{new}$ with the remaining terms in the error-state vector of the hybrid DEEP-EKF, we note that the IMU-state errors can be written as linear functions of the error-state, owing to the properties of the B-spline representation (see (2.22)-(2.24)):

$$\tilde{\mathbf{x}}_{I_k} = \boldsymbol{\Xi}_1 \tilde{\mathbf{x}}_k \tag{2.27}$$

$$\tilde{\mathbf{x}}_{I_{k+N}} = \begin{bmatrix} \boldsymbol{\Xi}_2 & \boldsymbol{\Xi}_3 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_{new} \\ \\ \tilde{\mathbf{x}}_k \end{bmatrix} \tag{2.28}$$

Using the above equations, along with (2.26), and solving for $\tilde{\mathbf{x}}_{new}$, we obtain:

$$\tilde{\mathbf{x}}_{new} = \boldsymbol{\Xi}_2^{\dagger} \left(\boldsymbol{\Phi}_I(t_{k+N}, t_k)\boldsymbol{\Xi}_1 - \boldsymbol{\Xi}_3 \right) \tilde{\mathbf{x}}_k + \boldsymbol{\Xi}_2^{\dagger} \mathbf{w}_k$$

where $\mathbf{\Xi}_2^\dagger$ is the pseudoinverse of $\mathbf{\Xi}_2$. Denoting $\mathbf{A} = \mathbf{\Xi}_2^\dagger \left(\mathbf{\Phi}_I(t_{k+N}, t_k)\mathbf{\Xi}_1 - \mathbf{\Xi}_3\right)$, we can now write the covariance matrix of the augmented error-state:

$$\mathbf{P}_{k+N|k} = \begin{bmatrix} \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}^T + \mathbf{\Xi}_2^\dagger \mathbf{Q}_k \mathbf{\Xi}_2^{\dagger T} & \mathbf{A}\mathbf{P}_{k|k} \\ \mathbf{P}_{k|k}\mathbf{A}^T & \mathbf{P}_{k|k} \end{bmatrix} \tag{2.29}$$

where $\mathbf{P}_{k|k}$ is the covariance matrix of the error-state vector after the update at time step k. As a final step, we remove from the error-state and the covariance matrix the terms corresponding to the IMU biases at time-step k, as these are no longer necessary.

2.5.4 Update

Once the error-state augmentation is complete, we proceed to process the feature measurements from the camera. Specifically, we process measurements in two ways:

- if a feature is no longer being tracked at time step $k+N$, and its feature track length is less than M, it is used to provide constraints between the last M poses in the sliding-window as well as the calibration parameters. The MSCKF method is employed in this case, which uses all the information from the measurements without including the feature into the state vector.

- if a feature is still being tracked at time step $k+N$, and its feature track length is larger than M, it is initialized into the state vector, and its subsequent measurements are used for update in an EKF-SLAM formulation.

MSCKF update

The update equations of the DEEP-MSCKF formulation follow closely those of the original MSCKF, with essentially the only difference being the way in which Jacobians

are computed. Considering a feature that has been observed in m images, the first step in the update process is to employ all its measurements to obtain an estimate of its position, ${}^{G}\hat{\mathbf{p}}_{f_i}$, via triangulation. Next, we compute the measurement residuals and the corresponding Jacobians, by following equations (2.13)-(2.15) in Section 2.4. Let us we denote the measurement residual expression of feature i from the j-th pose as:

$$\mathbf{r}_{ij} \simeq \mathbf{H}_{\boldsymbol{\theta}_{ij}} \mathbf{x}_{\boldsymbol{\theta}_{k+N}} + \mathbf{H}_{\mathbf{p}_{ij}} \mathbf{x}_{\mathbf{p}_{k+N}} + \mathbf{H}_{\mathbf{c}_{ij}} \tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}_{ij}} {}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{ij} \tag{2.30}$$

and by stacking all m residuals we obtain:

$$\mathbf{r}_{i} \simeq \mathbf{H}_{\boldsymbol{\theta}_{i}} \mathbf{x}_{\boldsymbol{\theta}_{k+N}} + \mathbf{H}_{\mathbf{p}_{i}} \mathbf{x}_{\mathbf{p}_{k+N}} + \mathbf{H}_{\mathbf{c}_{i}} \tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}_{i}} {}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{i} \tag{2.31}$$

where $\mathbf{r}_i$ is a block vector with elements $\mathbf{r}_{ij}$, $\mathbf{H}_{\boldsymbol{\theta}_i}$ is a matrix with block rows $\mathbf{H}_{\boldsymbol{\theta}_{ij}}$, $\mathbf{H}_{\mathbf{p}_i}$ is a matrix with block rows $\mathbf{H}_{\mathbf{p}_{ij}}$, $\mathbf{H}_{\mathbf{c}_i}$ is a matrix with block rows $\mathbf{H}_{\mathbf{c}_{ij}}$, $\mathbf{H}_{\mathbf{f}_i}$ is a matrix with block rows $\mathbf{H}_{\mathbf{f}_{ij}}$, and $\mathbf{n}_i$ is a block vector with elements $\mathbf{n}_{ij}$. Denoting $\mathbf{H}_{\mathbf{x}_i} = [\mathbf{0}\ \mathbf{H}_{\mathbf{c}_i}\ \mathbf{H}_{\mathbf{p}_i}\ \mathbf{H}_{\boldsymbol{\theta}_i}\ \mathbf{0}\ ...\ \mathbf{0}]$, (2.31) becomes:

$$\mathbf{r}_{i} \simeq \mathbf{H}_{\mathbf{x}_i} \tilde{\mathbf{x}}_{k+N} + \mathbf{H}_{\mathbf{f}_i} {}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{i} \tag{2.32}$$

Based on the linearized expression in (2.32), the update proceeds as in the original MSCKF, i.e., by removing ${}^{G}\tilde{\mathbf{p}}_{f_i}$ from the linearized expression. To achieve this, we define a matrix $\mathbf{V}_i$ whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$, and define $\mathbf{r}_i^o$ as:

$$\mathbf{r}_i^o = \mathbf{V}_i^T \mathbf{r}_i \simeq \mathbf{H}_i^o \tilde{\mathbf{x}}_{k+N} + \mathbf{n}_i^o \tag{2.33}$$

where $\mathbf{H}_i^o = \mathbf{V}_i^T \mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{n}_i^o = \mathbf{V}_i^T \mathbf{n}_i$. For computational efficiency, we can compute $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ without explicitly computing $\mathbf{V}_i$ [74]. Once $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ are computed, we proceed by performing a Mahalanobis gating test to reject outliers, and features whose residuals pass the test will be employed in the EKF update later.

SLAM update

The alternative way of processing the feature measurements is to include features into the state vector, and use their measurements as in a standard EKF-SLAM formulation. Specifically, the measurement residual expression of the i-th "SLAM" feature is the same as (2.32). However, in order to get a better linearity in the filter, the SLAM features employ an inverse-depth parameterization. More specifically, the i-th SLAM feature position is parametrized as:

$$^{G}\mathbf{p}_{f_i} = {}^{G}\mathbf{p}_i + \frac{1}{\rho_i}\mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ 1 \end{bmatrix} \tag{2.34}$$

where: (i) $\mathbf{f}_i = [\alpha_i \ \beta_i \ \rho_i]^T$ is the feature position in the inverse-depth parameterization, which is included in the state vector (see (2.19)). (ii) $^{G}\mathbf{p}_i$ is the IMU position of the state that is used as the "anchor" of the inverse-depth parameterization (this state is included in the estimated state vector). (iii) $\mathbf{R}_i$ is a constant rotation matrix. For the details of the inverse-depth parameterization, the reader is referred to [63]. Using this parameterization, the feature position error in frame $\{G\}$ becomes:

$$\begin{aligned} {}^{G}\tilde{\mathbf{p}}_{f_i} &= {}^{G}\tilde{\mathbf{p}}_i + \mathbf{H}_{i,IDP}\tilde{\mathbf{f}}_i \\ &= \mathbf{B}_{\mathbf{p}_i}\mathbf{x}_{\mathbf{p}_{k+N}} + \mathbf{H}_{i,IDP}\tilde{\mathbf{f}}_i \end{aligned} \tag{2.35}$$

where $\mathbf{B}_{\mathbf{p}_i}$ is the linear expression between the position error and position control points (see (2.23)). Substituting (2.35) in (2.32), we obtain the final measurement residual expression for the i-th SLAM feature:

$$\mathbf{r}_i \simeq \mathbf{H}_{s\mathbf{x}_i}\tilde{\mathbf{x}}_{k+N} + \mathbf{n}_i \tag{2.36}$$

where $\mathbf{H}_{s\mathbf{x}_i} = \begin{bmatrix}\mathbf{0} & \mathbf{H}_{\mathbf{c}_i} & (\mathbf{H}_{\mathbf{p}_i} + \mathbf{H}_{\mathbf{f}_i}\mathbf{B}_{\mathbf{p}_i}) & \mathbf{H}_{\boldsymbol{\theta}_i} & \mathbf{0} & \dots & (\mathbf{H}_{\mathbf{f}_i}\mathbf{H}_{i,IDP}) & \mathbf{0}\end{bmatrix}$.

Similar to the MSCKF update, we proceed by performing a Mahalanobis gating test to reject outliers, and features whose residuals pass the test are employed in the EKF update.

Once all the residuals from (2.33) and (2.36) are available, they are stacked together to form a residual vector:

$$\underbrace{\begin{bmatrix}\mathbf{r}_1^o \\ \vdots \\ \mathbf{r}_1 \\ \vdots\end{bmatrix}}_{\mathbf{r}} \simeq \underbrace{\begin{bmatrix}\mathbf{H}_1^o \\ \vdots \\ \mathbf{H}_{s\mathbf{x}_1} \\ \vdots\end{bmatrix}}_{\mathbf{H}} \tilde{\mathbf{x}}_{k+N} + \underbrace{\begin{bmatrix}\mathbf{n}_1^o \\ \vdots \\ \mathbf{n}_1 \\ \vdots\end{bmatrix}}_{\mathbf{n}} \tag{2.37}$$

and then used to carry out the EKF update:

$$\boldsymbol{\Delta}\mathbf{x} = \mathbf{K}\mathbf{r} \tag{2.38}$$

$$\mathbf{P}_{k+N|k+N} = \mathbf{P}_{k+N|k} - \mathbf{K}\mathbf{S}\mathbf{K}^T \tag{2.39}$$

$$\mathbf{S} = \mathbf{H}\mathbf{P}_{k+N|k}\mathbf{H}^T + \sigma^2\mathbf{I} \tag{2.40}$$

$$\mathbf{K} = \mathbf{P}_{k+N|k}\mathbf{H}^T\mathbf{S}^{-1} \tag{2.41}$$

Note that, unlike the calibration parameters and feature states whose correction can be directly applied, once the correction to the position and orientation knot parameters are

computed in an EKF update, we proceed to compute the pose corrections by employing (2.22)-(2.24), and subsequently these pose corrections are applied to the estimates in the state vector (2.19). After the update is complete, the oldest control points and the lost SLAM features are removed from the error-state vector, similarly to the original hybrid filter.

The most computationally expensive operation in the proposed EKF-based algorithm lies in the update, whose cost is approximately cubic in the size of the error-state vector, in the worst case. The error-state vector in the hybrid DEEP-EKF consists of three parts: the calibration parameters, B-spline control points and SLAM features. We therefore see that by changing the frequency at which new knots are introduced (i.e., changing N), the computational cost of the estimator will be significantly impacted. As knots are placed more sparsely (i.e., N is increased), the size of the error-state vector and the computational cost of the estimator will decrease. However, placing knots more sparsely also reduces the fidelity with which the trajectory can be described, and this degrades accuracy. The key result, which we experimentally demonstrate in Sections 2.7 and 2.8, is that while the reduction in computational cost is rapid as N increases, the loss of accuracy is only modest.

2.6 Observability Analysis

It has been shown that the performance of an EKF estimator in VIO can be significantly improved by computing the filter Jacobians in a way that ensures that the linearized system model has observability properties matching those of the underlying non-

linear system (i.e., the global position and yaw are unobservable) [64]. This can be achieved if the Jacobians are evaluated using a *single* estimate for each of the position and velocity states (the first available one). The resulting estimator, termed MSCKF 2.0, has improved accuracy and consistency[3]. We here follow a similar approach in the hybrid DEEP-EKF.

In this section, we first analyze the observability properties of the hybrid DEEP-EKF, and present the reason leading to inconsistency. Then, we provide a simple method to enforce the correct observability properties and present simulations to compare the DEEP formulations with and without enforcing the correct observability properties.

To simplify the derivation, we here carry out the analysis without including the IMU biases and calibration parameters. Moreover, the feature position error is defined in the global frame for simplicity. Instead of formulating the observability matrix for the system at hand and examining its rank, we here opt to examine the rank of the information matrix that expresses the information provided by all the measurements for estimating the error state. Specifically, this matrix is given by:

$$\begin{aligned}\mathbf{\Lambda} &= \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \\ &= \sum_k \mathbf{H}_{\omega_k}^T\mathbf{R}_{\omega_k}^{-1}\mathbf{H}_{\omega_k} + \sum_k \mathbf{H}_{\mathbf{a}_k}^T\mathbf{R}_{\mathbf{a}_k}^{-1}\mathbf{H}_{\mathbf{a}_k} + \sum_{i,k} \mathbf{H}_{\mathbf{z}_{ik}}^T\mathbf{R}_{\mathbf{z}_{ik}}^{-1}\mathbf{H}_{\mathbf{z}_{ik}}\end{aligned}$$

Where $\mathbf{H}_{\omega_k}$, $\mathbf{H}_{\mathbf{a}_k}$, and $\mathbf{H}_{\mathbf{z}_{ik}}$ represent the Jacobians corresponding to the accelerometer, gyroscope, and camera measurements respectively with respect to the "extended" error-state vector containing the entire history of states:

$$\tilde{\mathbf{x}}_\ell = [\underbrace{\mathbf{c}_{\mathbf{p}_\ell}^T\ \mathbf{c}_{\mathbf{p}_{\ell-N}}^T \cdots \mathbf{c}_{\mathbf{p}_0}^T}_{\mathbf{x}_{\mathbf{p}_\ell}^T}\ \underbrace{\mathbf{c}_{\boldsymbol{\theta}_\ell}^T\ \mathbf{c}_{\boldsymbol{\theta}_{\ell-N}}^T \cdots \mathbf{c}_{\boldsymbol{\theta}_N}^T}_{\mathbf{x}_{\boldsymbol{\theta}_\ell}^T}\ {}^G\tilde{\mathbf{p}}_{f_1}^T\ \cdots\ {}^G\tilde{\mathbf{p}}_{f_s}^T]^T \tag{2.42}$$

[3] A recursive estimator is termed consistent when the ensemble mean of the estimation errors is zero, and their ensemble covariance matrix is equal to the one reported by the estimator [2].

where $\mathbf{x}_{\mathbf{p}_\ell}$ includes the position control points that represent the position, velocity and acceleration errors, and similarly, $\mathbf{x}_{\boldsymbol{\theta}_\ell}$ includes the orientation control points that represent the orientation error and its derivative, and ${}^G\tilde{\mathbf{p}}_{f_i}$, $i = 1 \dots s$, is the i-th feature position error in the global frame. Note that $\tilde{\mathbf{x}}_\ell$ represents the error-states throughout the whole trajectory in the time interval $[0, t_\ell]$. Here we examine the rank of the matrix $\mathbf{\Lambda}$, in order to analyze the type of information that the sensors provide. To this end, we first examine the structure of the Jacobians shown in the expression for $\mathbf{\Lambda}$.

The IMU measurements are modeled as:

$$\boldsymbol{\omega}_m(t) = \boldsymbol{\omega}(t) + \mathbf{b}_\mathbf{g}(t) + \mathbf{n}_\mathbf{r}(t) \tag{2.43}$$

$$\mathbf{a}_m(t) = {}^I_G\mathbf{R}(t)\left({}^G\mathbf{a}(t) - \mathbf{g}\right) + \mathbf{b}_\mathbf{a}(t) + \mathbf{n}_\mathbf{a}(t) \tag{2.44}$$

where $\boldsymbol{\omega}(t)$ and ${}^G\mathbf{a}(t)$ denote the IMU angular rate and linear acceleration, $\mathbf{b}_\mathbf{g}(t)$ and $\mathbf{b}_\mathbf{a}(t)$ are the gyro and acceleration bias, respectively, $\mathbf{n}_\mathbf{r}(t)$ and $\mathbf{n}_\mathbf{a}(t)$ are white Gaussian noise processes, and $\mathbf{g}$ is the gravity vector in the global frame. By ignoring the biases and noises, we can express the IMU measurement residuals at time-step k as:

$$\tilde{\boldsymbol{\omega}}_m(t_k) \simeq {}^I_G\hat{\mathbf{R}}(t_k){}^G\dot{\tilde{\boldsymbol{\theta}}}_k \tag{2.45}$$

$$\tilde{\mathbf{a}}_m(t_k) \simeq {}^I_G\hat{\mathbf{R}}(t_k)\big({}^G\tilde{\mathbf{a}}_k + \lfloor\big({}^G\hat{\mathbf{a}}(t_k) - \mathbf{g}\big)\times\rfloor{}^G\tilde{\boldsymbol{\theta}}_k\big) \tag{2.46}$$

Thus, the Jacobians of the IMU measurements with respect to $\tilde{\mathbf{x}}_\ell$ are given by:

$$\mathbf{H}_{\boldsymbol{\omega}_k} = {}^I_G\hat{\mathbf{R}}(t_k)\begin{bmatrix}\mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{B}_2^{(1)}(t_k) & \mathbf{0}_3 & \cdots & \mathbf{0}_3\end{bmatrix} \tag{2.47}$$

$$\mathbf{H}_{\mathbf{a}_k} = {}^I_G\hat{\mathbf{R}}(t_k)\begin{bmatrix}\mathbf{B}_3^{(2)}(t_k) & \lfloor\big({}^G\hat{\mathbf{a}}(t_k) - \mathbf{g}\big)\times\rfloor\mathbf{B}_2(t_k) & \mathbf{0}_3 & \cdots & \mathbf{0}_3\end{bmatrix} \tag{2.48}$$

$$\mathbf{B}_2(t_k) = \begin{bmatrix}\mathbf{0}_3 \cdots & \alpha_1(t_k)\mathbf{I}_3 & \alpha_2(t_k)\mathbf{I}_3 & \alpha_3(t_k)\mathbf{I}_3 & \cdots \mathbf{0}_3\end{bmatrix} \tag{2.49}$$

$$\mathbf{B}_2^{(1)}(t_k) = \begin{bmatrix}\mathbf{0}_3 \cdots & \alpha_1^{(1)}(t_k)\mathbf{I}_3 & \alpha_2^{(1)}(t_k)\mathbf{I}_3 & \alpha_3^{(1)}(t_k)\mathbf{I}_3 & \cdots \mathbf{0}_3\end{bmatrix}$$

$$\mathbf{B}_3^{(2)}(t_k) = \begin{bmatrix}\mathbf{0}_3 \cdots & \beta_1^{(2)}(t_k)\mathbf{I}_3 & \beta_2^{(2)}(t_k)\mathbf{I}_3 & \beta_3^{(2)}(t_k)\mathbf{I}_3 & \beta_4^{(2)}(t_k)\mathbf{I}_3 & \cdots \mathbf{0}_3\end{bmatrix}$$

where $\mathbf{B}_j^{(i)}(t_k)$ denotes the i-th order derivative of a j-degree B-spline function at time-step k, e.g., $\mathbf{B}_3^{(2)}(t_k)$ is the Jacobian of the 2-nd order derivative of the position error at t_k with respect to the position control points $\mathbf{x}_{\mathbf{p}_\ell}$, and similarly, $\mathbf{B}_2(t_k)$ and $\mathbf{B}_2^{(1)}(t_k)$ denote Jacobians of the orientation error and orientation error derivative at t_k with respect to the orientation control points $\mathbf{x}_{\boldsymbol{\theta}_\ell}$, respectively. α_i, $\alpha_i^{(1)}$ and $\beta_i^{(2)}$ are the coefficients of the corresponding B-spline function, and they have the following properties:

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

$$\alpha_1^{(1)} + \alpha_2^{(1)} + \alpha_3^{(1)} = 0$$

$$\beta_1^{(2)} + \beta_2^{(2)} + \beta_3^{(2)} + \beta_4^{(2)} = 0$$

Assuming a calibrated perspective camera is used, the measurement of a feature i at time-step k is modelled as:

$$\mathbf{z}_{ik} = \mathbf{h}(\boldsymbol{\pi}_k, {}^G\mathbf{p}_{f_i}) + \mathbf{n}_{ik} = \mathbf{h}({}^{C_k}\mathbf{p}_{f_i}) + \mathbf{n}_{ik} = \begin{bmatrix}\frac{{}^{C_k}x_{f_i}}{{}^{C_k}z_{f_i}} \\ \frac{{}^{C_k}y_{f_i}}{{}^{C_k}z_{f_i}}\end{bmatrix} + \mathbf{n}_{ik} \tag{2.50}$$

where $\mathbf{n}_{ik}$ is zero-mean white Gaussian noise with covariance matrix $\sigma^2\mathbf{I}_2$, ${}^{C_k}\mathbf{p}_{f_i} = \begin{bmatrix} {}^{C_k}x_{f_i} & {}^{C_k}y_{f_i} & {}^{C_k}z_{f_i} \end{bmatrix}^T$ is the feature position in the camera frame at time-step k, and

$$ {}^{C_k}\mathbf{p}_{f_i} = {}^{C}_{I}\mathbf{R}\ {}^{I_k}_{G}\mathbf{R}\big({}^{G}\mathbf{p}_{f_i} - {}^{G}\mathbf{p}_{I_k}\big) + {}^{C}\mathbf{p}_{I} \tag{2.51} $$

If feature i is processed at time-step $\ell_i + 1$ (e.g., due to losing track at time-step $\ell_i + 1$), the the Jacobian of (2.50) with respect to $\tilde{\mathbf{x}}_\ell$ is given by:

$$ \mathbf{H}_{\mathbf{z}_{ik}} = \mathbf{H}_{\mathbf{f}_{ik}}(\hat{\boldsymbol{\pi}}_{k|\ell_i}, {}^{G}\hat{\mathbf{p}}_{f_i}) \begin{bmatrix} -\mathbf{B}_3(t_k) & \lfloor\big({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{k|\ell_i}\big)\times\rfloor\mathbf{B}_2(t_k) & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{bmatrix} \tag{2.52} $$

$$ \mathbf{H}_{\mathbf{f}_{ik}}(\hat{\boldsymbol{\pi}}_{k|\ell_i}, {}^{G}\hat{\mathbf{p}}_{f_i}) = \mathbf{J}_{ik}(\hat{\boldsymbol{\pi}}_{k|\ell_i}, {}^{G}\hat{\mathbf{p}}_{f_i})\ {}^{C}_{I}\mathbf{R}\ {}^{I}_{G}\hat{\mathbf{R}}_{k|\ell_i} \tag{2.53} $$

$$ \mathbf{J}_{ik}(\hat{\boldsymbol{\pi}}_{k|\ell_i}, {}^{G}\hat{\mathbf{p}}_{f_i}) = \frac{1}{{}^{C_i}\hat{z}_{f_i}} \begin{bmatrix} 1 & 0 & \frac{-{}^{C_i}\hat{x}_{f_i}}{{}^{C_i}\hat{z}_{f_i}} \\ 0 & 1 & \frac{-{}^{C_i}\hat{y}_{f_i}}{{}^{C_i}\hat{z}_{f_i}} \end{bmatrix} \tag{2.54} $$

$$ \mathbf{B}_3(t_k) = \begin{bmatrix} \mathbf{0}_3 \cdots & \beta_1(t_k)\mathbf{I}_3 & \beta_2(t_k)\mathbf{I}_3 & \beta_3(t_k)\mathbf{I}_3 & \beta_4(t_k)\mathbf{I}_3 & \cdots \mathbf{0}_3 \end{bmatrix} \tag{2.55} $$

where $\mathbf{B}_2(t_k)$ is defined in (2.49), and $\mathbf{B}_3(t_k)$ is the Jacobian of the position error at t_k with respect to the position control points $\mathbf{x}_{\mathbf{p}_\ell}$. Due to the properties of B-splines, the following equation holds:

$$ \beta_1 + \beta_2 + \beta_3 + \beta_4 = 1 $$

2.6.1 "Ideal" Observability Properties

It is interesting to first examine the "ideal" case, where all the Jacobians are evaluated at the *true* states and the *true* trajectory matches the corresponding B-spline functions during the time interval $[0, t_\ell]$. We can then define a matrix $\mathbf{N}_{null}$ as:

$$\mathbf{N}_{null} = \begin{bmatrix} \mathbf{N}_1 & \mathbf{N}_2 \end{bmatrix} \tag{2.56}$$

$$= \begin{bmatrix} \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{I}_3 \\ \vdots \\ \mathbf{I}_3 \\ \mathbf{0}_3 \\ \vdots \\ \mathbf{0}_3 \\ \mathbf{I}_3 \\ \vdots \\ \mathbf{I}_3 \end{bmatrix} & \begin{bmatrix} -\lfloor \boldsymbol{\gamma}_\ell \times \rfloor \mathbf{g} \\ -\lfloor \boldsymbol{\gamma}_{\ell-N} \times \rfloor \mathbf{g} \\ \vdots \\ -\lfloor \boldsymbol{\gamma}_0 \times \rfloor \mathbf{g} \\ \mathbf{g} \\ \vdots \\ \mathbf{g} \\ -\lfloor {}^G\mathbf{p}_{f_1} \times \rfloor \mathbf{g} \\ \vdots \\ -\lfloor {}^G\mathbf{p}_{f_s} \times \rfloor \mathbf{g} \end{bmatrix} \end{bmatrix} \tag{2.57}$$

where $\boldsymbol{\gamma}_j$ is the vector containing the set of position control points that are used to represent the j-th position. Note that these control points are different than the control points shown in (2.42), since the latter represent the *errors* of the estimates, while the $\boldsymbol{\gamma}_j$ control points represent the trajectory itself.

If we define $\boldsymbol{\gamma}_{\mathbf{p}\ell} = \begin{bmatrix} \boldsymbol{\gamma}_\ell^T & \boldsymbol{\gamma}_{\ell-N}^T & \cdots & \boldsymbol{\gamma}_0^T \end{bmatrix}^T$ to be the vector containing all the control points used to represent the position trajectory up to time-step ℓ, then:

$$^G\hat{\mathbf{a}}_B(t_k) = \mathbf{B}_3^{(2)}(t_k)\boldsymbol{\gamma}_{\mathbf{p}\ell} \tag{2.58}$$

$$^G\hat{\mathbf{p}}_B(t_k) = \mathbf{B}_3(t_k)\boldsymbol{\gamma}_{\mathbf{p}\ell} \tag{2.59}$$

where $^G\hat{\mathbf{a}}_B(t_k)$ and $^G\hat{\mathbf{p}}_B(t_k)$ are the acceleration and position estimates at time-step k computed using the same position control points ($\boldsymbol{\gamma}_{\mathbf{p}\ell}$), respectively. In the "ideal" case, the position and acceleration computed using the position control points are equal to the *true* states. Thus, it is easy to verify that in this case the following equations hold:

$$\mathbf{H}_{\boldsymbol{\omega}_k} \cdot \mathbf{N}_{null} = \mathbf{0}_{3\times 4} \tag{2.60}$$

$$\mathbf{H}_{\mathbf{a}_k} \cdot \mathbf{N}_{null} = \mathbf{0}_{3\times 4} \tag{2.61}$$

$$\mathbf{H}_{\mathbf{z}_{ik}} \cdot \mathbf{N}_{null} = \mathbf{0}_{2\times 4} \tag{2.62}$$

Since these equations hold for any i and k, and the four columns of $\mathbf{N}_{null}$ are linearly independent, we conclude that these form a basis for the nullspace of *all* the measurement Jacobian matrices. In turn this means that these columns also form a basis for the nullspace of $\boldsymbol{\Lambda}$. This nullspace matrix is similar to the nullspace of the observability matrix in [64], and they share the same intuition for the nature of the unobservable subspace. Intuitively, the first three columns of $\mathbf{N}_{null}$ ($\mathbf{N}_1$) represent the global translation of the state vector (i.e., shifting all position control points by the same amount), and the last column of $\mathbf{N}_{null}$ ($\mathbf{N}_2$) corresponds to the yaw (the rotation along the gravity) [61]. Also, since the rank of $\mathbf{N}_{null}$ is four, the observability properties in this case match those of the nonlinear model (i.e., they indicate an unobservable global position and yaw), as desired.

2.6.2 Observability in the DEEP formulation

In the real-world case, when *estimates* are used to compute the Jacobian matrices (instead of the true values), equations (2.61) and (2.62) do not hold. Instead, we obtain:

$$\mathbf{H}_{\mathbf{a}_k} \cdot \mathbf{N}_1 = \mathbf{0}_{3\times 3} \tag{2.63}$$

$$\mathbf{H}_{\mathbf{z}_{ik}} \cdot \mathbf{N}_1 = \mathbf{0}_{2\times 3} \tag{2.64}$$

$$\mathbf{H}_{\mathbf{a}_k} \cdot \mathbf{N}_2 = {}^{I}_{G}\hat{\mathbf{R}}(t_k)\lfloor\big({}^{G}\hat{\mathbf{a}}(t_k) - \mathbf{g} - {}^{G}\hat{\mathbf{a}}_B(t_k)\big)\times\rfloor\mathbf{g} \tag{2.65}$$

$$\mathbf{H}_{\mathbf{z}_{ik}} \cdot \mathbf{N}_2 = \mathbf{H}_{\mathbf{f}_{ik|\ell_i}}\lfloor\big({}^{G}\hat{\mathbf{p}}_B(t_k) - {}^{G}\hat{\mathbf{p}}_{k|\ell_i}\big)\times\rfloor\mathbf{g} \tag{2.66}$$

We can thus see that $\mathbf{N}_2$ no longer belongs in the nullspace of all the measurement Jacobians. As a result, the nullspace of measurement Jacobians is only of rank three in general, spanned by $\mathbf{N}_1$. This means that the global yaw erroneously *appears to be* observable. As a result, the estimator underestimates the covariance of the yaw, as we can see in Section 2.6.3.

To address this problem, we need to make changes in the way Jacobians are computed, to ensure that the dimension of the nullspace of the information matrix is "restored" to four. We start by noting that (2.65) and (2.66) are non-zero for two reasons:

1. Different estimates of the same IMU states may be used when evaluating the camera-measurement Jacobians. This is due to the fact that the same IMU states may be involved in processing measurements at different time-steps ℓ_i.

2. Recall that the control points appearing in (2.57) describe a B-spline representation of the *true trajectory*. The fact that all the Jacobians have a common nullspace of dimension four (i.e., the fact that (2.57) holds) depends crucially on the true trajectory being represented by a B-spline. By contrast, in the DEEP formulation, only the errors

are assumed to follow a B-spline representation, while the trajectory estimates can be arbitrary. This means that there is no B-spline which can ensure that (i) ${}^{G}\hat{\mathbf{a}}(t_k)$ equals ${}^{G}\hat{\mathbf{a}}_B(t_k)$ for all t_k, and (ii) ${}^{G}\hat{\mathbf{p}}_B(t_k)$ equals ${}^{G}\hat{\mathbf{p}}_{k|\ell_i}$ for all t_k at the same time. These problems become more severe as N increases, since in that case the difference between the trajectory estimate and a hypothetical "best-fit" B-spline will be larger. In order to ensure that (2.65) and (2.66) are zero, for any time interval $[t_{k-N}, t_k]$, the following equations must hold:

$$\begin{cases} {}^{G}\hat{\mathbf{a}}_B(t_{k-N+1}) & = {}^{G}\hat{\mathbf{a}}(t_{k-N+1}) \\ {}^{G}\hat{\mathbf{p}}_B(t_{k-N+1}) & = {}^{G}\hat{\mathbf{p}}_{k-N+1|\ell_i} \\ & \vdots \\ {}^{G}\hat{\mathbf{a}}_B(t_k) & = {}^{G}\hat{\mathbf{a}}(t_k) \\ {}^{G}\hat{\mathbf{p}}_B(t_k) & = {}^{G}\hat{\mathbf{p}}_{k|\ell_i} \end{cases} \tag{2.67}$$

where ${}^{G}\hat{\mathbf{a}}_B$ and ${}^{G}\hat{\mathbf{p}}_B$ are computed from (2.58) and (2.59) using the same set of control points representing the trajectory ($\boldsymbol{\gamma}_{\mathbf{p}\ell}$). Equation (2.67) means that the position and acceleration estimates involved in computing Jacobians must conform to a B-spline representation, which is not satisfied for general trajectories.

2.6.3 Enforcing consistency

In order to ensure the correct observability properties in the hybrid DEEP-EKF, we can use the first available estimate of the position control points to compute Jacobians. Specifically, prior to each state augmentation and update we perform a local B-spline fitting of the new segment of the trajectory, and use the resulting B-spline estimates to compute

the position and velocity estimates during this interval. The computed position and velocity quantities are then used in evaluating all Jacobians involving them. By doing this, the position, velocity and acceleration estimates used in Jacobian evaluation conform to a B-spline model of the trajectory. Therefore, the requirement (2.67) is satisfied, leading to (2.61) and (2.62). We stress that, similarly to [64], these estimates are only used in order to evaluate the Jacobians, and are not used in computing residuals. As a result of this modification, the hybrid DEEP-EKF estimator remains consistent.

Here we present the results from 100 Monte-Carlo simulations to compare the performance of the DEEP formulation with and without enforcing the dimension of the unobservable subspace to equal four. The simulation environment is the same as the one described in Section 2.7.

Before showing the full results from Monte-Carlo simulations, it is interesting to examine the results from a single representative trial. Fig. 2.2 shows the orientation estimation errors of the yaw axis and its $\pm 3\sigma$ envelopes from the reported covariances (which corresponds to 99.7% confidence region). As we can see in the figure, the reported yaw standard deviation of the DEEP formulation when Jacobians are computed without any modification fluctuates as time goes on, erroneously indicating the yaw being observable. On the other hand, when the Jacobians are modified as described in the preceding subsection, the reported yaw standard deviation decreases over time, showing that the yaw is unobservable, as desired.

We now present the results from 100 Monte-Carlo simulations. Table 2.1 provides detailed numerical values for the NEES and RMSE with N ranging from 1 to 5. These

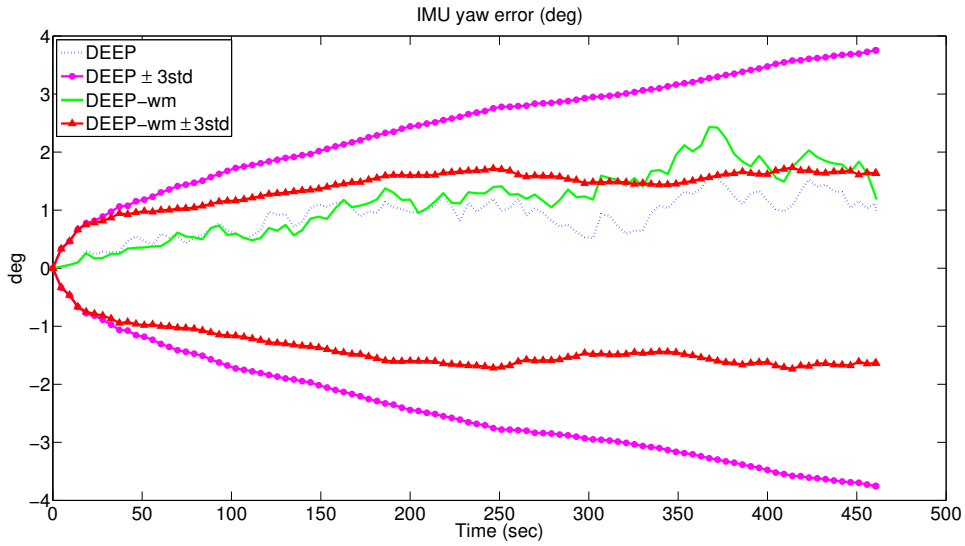

Figure 2.2: IMU yaw error and $\pm 3\sigma$ envelopes in one representative trial. The yaw error for the DEEP-EKF with modified Jacobians to ensure consistency (dashed line - blue), and without any modification (solid line - green). The $\pm 3\sigma$ envelopes for the DEEP-EKF with modified Jacobians (circles - magenta), and without (triangles - red).

results show that by enforcing the correct observability properties, the DEEP formulation obtains both improved consistency (the average NEES is closer to 6), and higher accuracy.

2.7 Simulations

In this section we present the results from two sets of Monte-Carlo simulations to demonstrate the performance of the DEEP formulation of the hybrid EKF. In our simulator, we generate trajectories emulating a platform (e.g., a handheld device) moving in a building-sized environment, in a trajectory that involves significant rotations and accelerations. In each Monte-Carlo trial a 7.5-minute, approximately 620-m long trajectory is generated, as

Table 2.1: Simulations: Comparison between the DEEP formulations with the proposed Jacobian modification (DEEP), and without modification (DEEP-wm).

	N=1		N=3		N=5	
	DEEP-wm	DEEP	DEEP-wm	DEEP	DEEP-wm	DEEP
Orientation RMSE (deg)	0.554	0.531	0.573	0.530	0.634	0.527
Position RMSE (m)	0.510	0.495	0.516	0.491	0.596	0.498
Pose NEES	6.28	6.10	6.29	5.99	9.23	6.07

well as feature tracks and inertial measurements with characteristics matching those that we observe in real-world datasets. Specifically, the images are available at 20 Hz, IMU measurements at 100 Hz, and the average feature-track length is 7.4 frames. In each trial, different feature positions and different noise realizations are used. The noise parameters match those of the sensors found on a LG Nexus 4 mobile device.

2.7.1 DEEP formulation vs. the original hybrid EKF

We begin by comparing the proposed DEEP formulation of the hybrid EKF to the "standard", pose-based formulation in [62]. Since the hybrid DEEP-EKF is derived by employing an approximate representation of the errors, we expect the original hybrid EKF to attain the highest estimation accuracy. As we discussed in Section 2.4, there can be different choices on the order of Taylor-series kept in the hybrid EKF for rolling-shutter

measurements (see (2.11) and (2.12)), which leads to different estimation accuracies and computational costs. Here we compare two of them: (i) $I_{\mathbf{p}} = 1$ and $I_{\boldsymbol{\theta}} = 1$, which means we keep up to the 1-st-order of Taylor-series of the pose errors, and thus the velocity and angular velocity are included in the state vector. (ii) $I_{\mathbf{p}} = 0$ and $I_{\boldsymbol{\theta}} = 0$, which means we only keep the 0-th-order of Taylor-series of the pose errors, and only the position and orientation are included for each pose in the state vector. We also employed a "keyframe" approach in which we only process measurements whose poses have "large enough" motion[4], which achieves smaller computational cost by reducing the size of the estimated state vector. In this way, we can compare the performance of our approach to that obtained when simply discarding information. We compare the results of the hybrid EKF to its DEEP formulation with varying N. Our goal is to examine the trade-off between the loss of accuracy and the computational gains as N changes, and compare to that of the original hybrid approaches. All estimators process the same data, and the sliding-window length at each time instant is equal to the current longest feature track (with a maximum allowable length corresponding to 0.75 sec, or 15 poses).

The performance metrics we compute are (i) the root-mean-squared errors (RMSE) for the position and orientation, (ii) the computational cost of the methods, expressed in terms of the filter runtime in the augmentation/update cycle per image measurement[5], and (iii) the normalized estimation error squared (NEES) for the pose errors. If we denote the

[4]Here we set the standard of the "large enough" motion as: the displacement between two frames is larger than 0.08m, or the rotation is larger than 2.5 deg. This choice is not unique. However, a larger threshold results in losing more information, which, as we will show later, leads to larger pose error.

[5]The timing results are obtained on a laptop computer with a Intel Core i7-3770 CPU at 3.40GHz, and the filter is implemented using a single-thread C++ code. The measured time contains all processing in the augmentation/update cycle, but it does *not* include IMU propagation, rolling-shutter motion computations, since they are always the same for both formulations.

pose error at time step k as $\tilde{\mathbf{x}}_p(k)$ and the corresponding 6×6 covariance matrix reported by the filter by $\mathbf{P}_p(k)$, the NEES at this time instant is defined as $\tilde{\mathbf{x}}_p(k)^T\mathbf{P}_p^{-1}(k)\tilde{\mathbf{x}}_p(k)$. Examining the NEES gives an insight into the magnitude of the unmodeled errors incurred by the DEEP parameterization. Specifically, if significant unmodeled errors exist, the covariance matrix reported by the estimator will be smaller than the covariance matrix of the actual errors (i.e., the estimator will be inconsistent [2]), and the NEES will increase. In a consistent estimator, the expected value of the pose NEES should equal six. Thus, by examining the deviation of the average NEES from this value, we can evaluate the significance of the unmodeled errors.

Fig. 2.3 shows the RMSE as well as the pose NEES for the different methods, averaged over 100 Monte-Carlo trials. In this plot, we compare the hybrid EKF method to the hybrid DEEP approaches, using values for the "knot-spacing factor" $N = 1, 5, 10, 15$. From these plots, we can observe that up to $N = 10$, the average performance of the DEEP formulation is almost indistinguishable to that of the pose-based hybrid formulation, while the estimation accuracy is noticeably lower for $N = 15$. Also, the differences between different $I_{\mathbf{p}}, I_{\boldsymbol{\theta}}$ in the hybrid EKF are almost indistinguishable, while the RMSE is much larger for the"keyframe" method. To explore these results in more detail, in Table 2.2 we show the average RMSE and NEES for all algorithms, averaged over all Monte-Carlo trials and all time. Moreover, in this table we include the average runtime per image update for the different algorithms.

We can draw three key conclusions from the results of Table 2.2:

- $I_{\mathbf{p}}$ and $I_{\boldsymbol{\theta}}$ have very small effect on the accuracy of device pose estimation, which matches the results from [62].

- As the knot density decreases in the hybrid DEEP-EKF (i.e., as N increases), the computational cost of the algorithm decreases rapidly, while the estimation errors increase slowly. For instance, when $N = 5$, the position error is 1.7% higher compared to the "best" hybrid EKF ($I_{\mathbf{p}} = 1$ and $I_{\boldsymbol{\theta}} = 1$), but the runtime is reduced by 84%. Even compared to the low-dimensional approximation in the pose-based EKF ($I_{\mathbf{p}} = 0$ and $I_{\boldsymbol{\theta}} = 0$), the accuracy in the DEEP formulation is less than 1% worse, while still achieving 60% runtime savings.

- If we compare to the keyframe algorithm, which has similar runtime to the DEEP formulation when $N = 5$, the position RMSE in the keyframe approach is 20% larger than the later. This shows that the error incurred by B-spline assumption is smaller than the effect of losing information in the filter.

Moreover, note that for values up to $N = 10$, the NEES remains close to the "ideal" value of 6. This indicates that the unmodelled errors resulting from the DEEP formulation are sufficiently small.

Since we are performing online calibration, it is useful to examine the results of the calibration parameters, shown in Fig. 2.4. These results show the parameter RMSE reported from different filters over 100 Monte-Carlo simulations. Similar to Fig. 2.3, the results in the hybrid DEEP-EKF are similar until $N = 10$. However, it is worth noting that the RMSE of some parameters from the hybrid filter with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ is significantly larger

Table 2.2: Simulation results: hybrid EKF vs. hybrid DEEP-EKF for varying N

	Hybrid EKF			Hybrid DEEP-EKF			
	$I_{\mathbf{p}} = 1$, $I_{\boldsymbol{\theta}} = 1$	$I_{\mathbf{p}} = 0$, $I_{\boldsymbol{\theta}} = 0$	$I_{\mathbf{p}} = 0$, $I_{\boldsymbol{\theta}} = 0$ keyframe	N=1	N=5	N=10	N=15
Orientation RMSE (deg)	0.804	0.807	0.978	0.825	0.837	0.854	0.919
Position RMSE (m)	0.946	0.953	1.155	0.962	0.962	1.023	1.094
Pose NEES	6.22	6.64	6.37	6.31	6.32	6.8	7.78
Runtime per image (msec)	2.0337	0.7996	0.3300	1.1298	0.3169	0.2696	0.2508

than other methods (e.g. the vertical coordinate of the principal point). This is caused by the truncation error that does not include velocity or angular velocity in approximating the pose errors, which also degrades the IMU pose accuracy compared to the higher-order approximations. The DEEP formulation, on the other hand, naturally includes the information of the velocity and angular velocity, and thus achieves similar results to the hybrid filter with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$, at a much lower computational cost.

2.7.2 DEEP formulation vs. B-spline trajectory parameterization

We next compare the performance of the proposed DEEP formulation to a version of the hybrid EKF estimator that employs the temporal basis function (TBF) formulation of [27, 68, 79]. The latter estimator is similar to the one presented in Section 2.5, with the difference that the trajectory itself, rather than just the errors, are modeled by B-splines. This comparison is informative as both these approaches have practically identical

computational cost, since for the same value of N they involve error-state vectors of the same size.

Table 2.3 shows the results of 100 Monte-Carlo simulations, comparing the two approaches for N ranging between one and five[6]. We can observe that the two parameterizations have similar results when $N = 1$, but the performance of the TBF approach degrades rapidly for larger values of N. When $N = 5$, the position RMSE of the TBF formulation is more than quadruple that of the DEEP. Moreover, the NEES of the TBF formulation is larger than its ideal value by an order of magnitude.

Two important conclusions can be drawn from these results. First, it becomes clear that the good performance of the DEEP approach cannot be attributed to the use of a "smooth" trajectory in the simulator. If that were the case, then a B-spline representation of the trajectory (which is used in the TBF approach) would suffice, and would also yield good results. Second, we can see that the decoupling of the representation of trajectory *estimates* from that of the trajectory *errors*, which is the key novelty of the DEEP formulation, yields added representational power, compared to a standard B-spline representation of the trajectory. Since the trajectory estimates are represented by a set of poses, no assumption on the form of the trajectory itself is imposed, and this leads to the favorable accuracy-computation trade-off observed in Table 2.2.

[6]For this comparison we only use the first 3 min measurements.

Table 2.3: Simulation results: DEEP vs. TBF formulation

	N=1		N=3		N=5	
	DEEP	TBF	DEEP	TBF	DEEP	TBF
Orientation RMSE (deg)	0.569	0.563	0.563	0.577	0.572	0.821
Position RMSE (m)	0.491	0.491	0.485	0.631	0.501	2.166
NEES	6.24	6.40	6.16	11.66	6.24	67.86

2.8 Real-World Experiments

We next present the results of two real-world experiments, one with a global-shutter camera and one with a rolling-shutter one, to show the capability of our DEEP algorithm to handle both cases. Shi-Tomasi features were extracted from images [88], and normalized cross-correlation was used for feature matching. All feature tracks were extracted in a pre-processing step, and used by all the algorithms compared, to ensure that all methods use exactly the same measurements.

2.8.1 Experiment with a global-shutter camera

We first present the results of a real-world experiment, in which a camera/IMU platform was mounted on a car driven in a residential area of Riverside, CA. The trajectory

length was 5km, driven in 20 min, and sample images from the experiment are shown in Fig. 2.6 [111].

Fig. 2.5 shows the trajectory estimated by the hybrid EKF and the hybrid DEEP-EKF with N ranging from 1 to 15. The trajectory is plotted on a map of the area, along with the ground-truth trajectory provided by a high-precision GPS/INS solution. Moreover, in Fig. 2.7 we plot the position errors (distance between the computed estimates and ground truth) throughout the trajectory. Table 2.4 lists the average position errors throughout the trajectory. In fact, for this specific dataset, the smallest average errors are obtained by the hybrid DEEP-EKF with $N = 15$. This is due to the stochastic nature of the noise, and we expect that if several experiments were conducted, the accuracy would decrease monotonically with N, as in the Monte-Carlo simulations.

Table 2.4: Real-world experiment: hybrid DEEP-EKF vs. hybrid EKF

	Hybrid EKF	N=1	N=5	N=10	N=15
Avg. position error (m)	13.81	14.73	15.32	14.61	11.88
Runtime (msec)	4.3631	9.1628	1.9588	1.2365	0.9844

Table 2.5: Real-world experiment 2: hybrid DEEP-EKF vs. hybrid EKF

	Hybrid EKF		Hybrid DEEP-EKF			
	$I_{\mathbf{p}} = 1$, $I_{\boldsymbol{\theta}} = 1$	$I_{\mathbf{p}} = 0$, $I_{\boldsymbol{\theta}} = 0$	N=1	N=5	N=10	N=15
Avg. position difference (m)	0	0.51	0.73	1.86	2.33	3.75
Final position error (m)	5.05	5.36	4.13	4.38	2.75	3.13
Runtime (msec)	2.7254	1.1769	1.8624	0.5664	0.4585	0.4496

2.8.2 Experiment with a rolling-shutter camera

We now present the results of another real-world experiment, in which a Nexus 4 cellphone was hand-held by a person walking in an indoor area of the UCR Bourns Hall. The trajectory length is about 690m, lasting 8.6 min, and sample images from the experiment are shown in Fig. 2.10.

Fig. 2.8 shows the trajectory estimated by the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$ and $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$, as well as the hybrid DEEP-EKF with N ranging from 1 to 15 [7]. Since this is an indoor experiment, we do not have access access to an accurate ground-truth trajectory by using GPS. However, as we can see in Section 2.7.1, the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$ attains the highest accuracy, thus here we treat its estimate as the "ground-truth" and use it to compute the estimated position differences of the DEEP-EKF with varying N, shown in Fig. 2.9. Moreover, since we know that the trajectory starts and ends at the same position, we can compute the final position error. The detailed numerical comparison is

[7] Since the keyframe approach attains a much lower estimation accuracy than other methods, it is not considered in the experiment.

listed in Table 2.5, as well as the computational cost of all algorithms, measured as the runtime per image.

We can observe that, as in the simulation results presented in the preceding section, the hybrid DEEP formulation results in accuracy that is similar to that of the original hybrid algorithm, but has significantly lower computational cost when $N \geq 5$. For this specific dataset, the minimum final position error is achieved for the hybrid DEEP-EKF when $N = 10$. This is due to the stochastic nature of the noise, and we expect that if several experiments were conducted, the accuracy would decrease monotonically with N , as in the Monte-Carlo simulations.

2.9 Conclusion

We have presented a novel formulation for the representation of the trajectory in pose-estimation problems. The key idea of the proposed DEEP approach is the decoupling of the representation of the trajectory *estimates* from that of the trajectory *errors*. Specifically, for the former a general, pose-based representation is employed, which imposes no assumptions on the form of the trajectory. On the other hand, for the latter a B-spline representation is used. This makes it possible to reduce the dimensionality of an estimator's error-state vector, simply by lowering the frequency at which knots are introduced. This general approach is used to design a novel VIO algorithm, which we term the hybrid DEEP-EKF. We applied this DEEP formulation to the 3D localization problem using rolling-shutter measurements, along with online calibration. By comparing the performance of this method to the "traditional" pose-based hybrid filter formulation of [62], we demon-

strate that the DEEP formulation yields substantial gains in computational efficiency, while resulting in only small loss of accuracy.

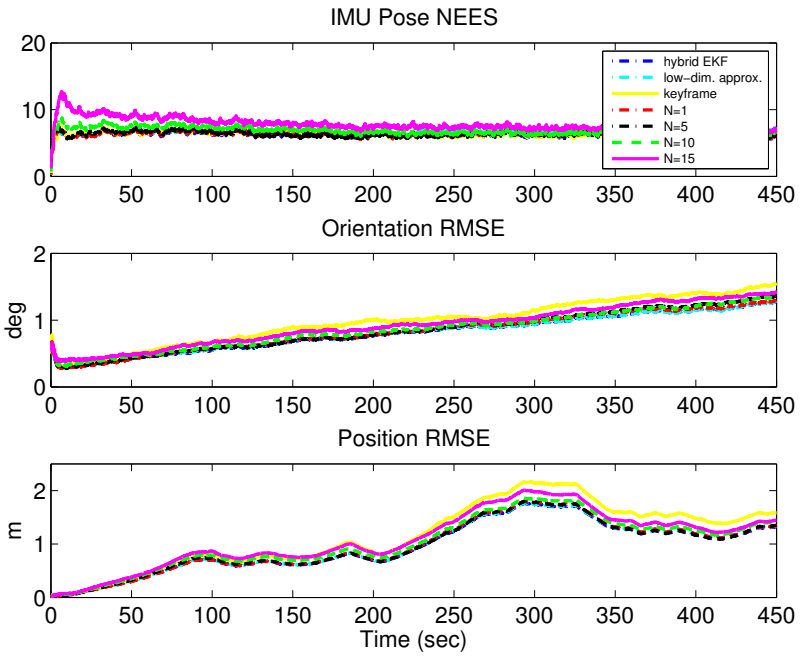

Figure 2.3: Simulation results: the position and orientation RMSE, as well as the IMU-pose NEES over time. Plots are averages over 100 Monte-Carlo trials. The compared methods are: the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$ (blue dash-dotted line), $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (cyan dash-dotted line), and the keyframe approach with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (yellow solid line); the hybrid DEEP-EKF with $N = 1$ (red dash-dotted line), $N = 5$ (black dash-dotted line), $N = 10$ (green dashed line), and $N = 15$ (magenta solid line).

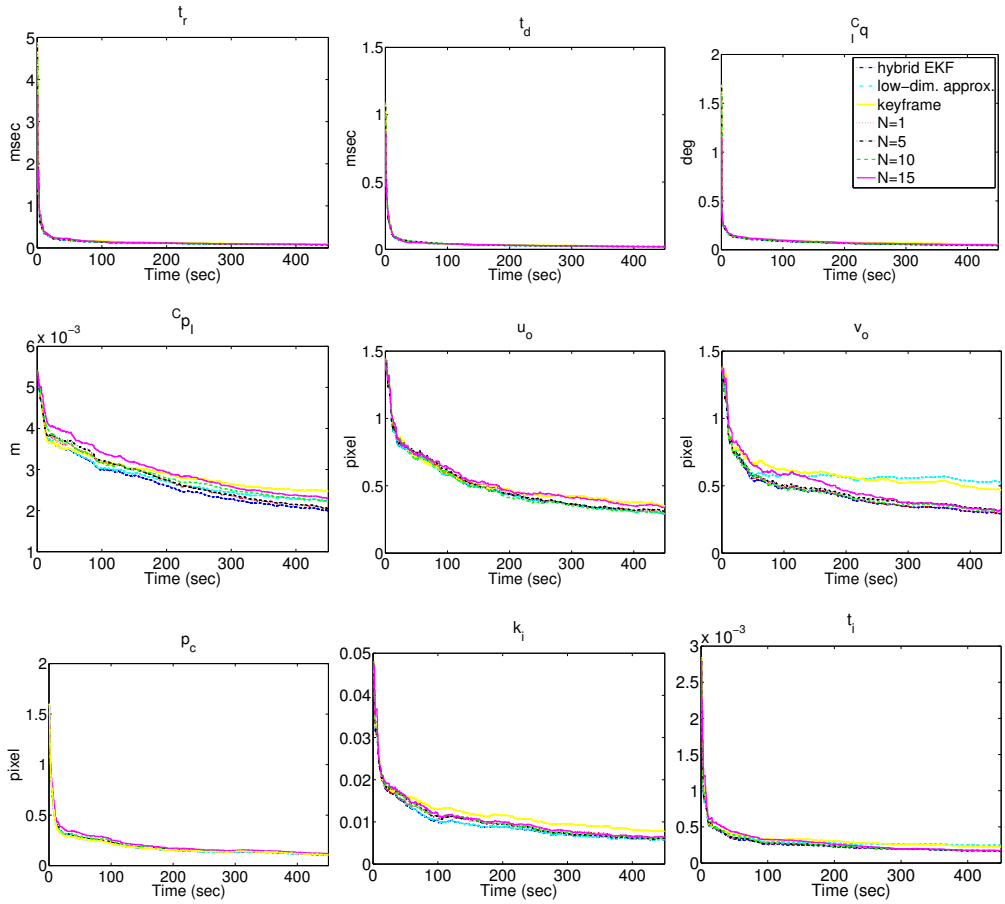

Figure 2.4: Parameter calibration results: the RMSE over 100 Monte-Carlo simulations. (Left to right, top to bottom) (a) rolling-shutter readout time, (b) camera-to-IMU time offset, (c) camera-to-IMU rotation, (d) camera-to-IMU translation, (e) horizontal principal point coordinate, (f) vertical principal point coordinate, (g) camera focal length, (h) camera radial distortion, (i) camera tangential distortion. The compared methods are: the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$ (blue dash-dotted line), $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (cyan dash-dotted line), and the keyframe approach with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (yellow solid line); the hybrid DEEP-EKF with $N = 1$ (red dash-dotted line), $N = 5$ (black dash-dotted line), $N = 10$ (green dashed line), and $N = 15$ (magenta solid line).

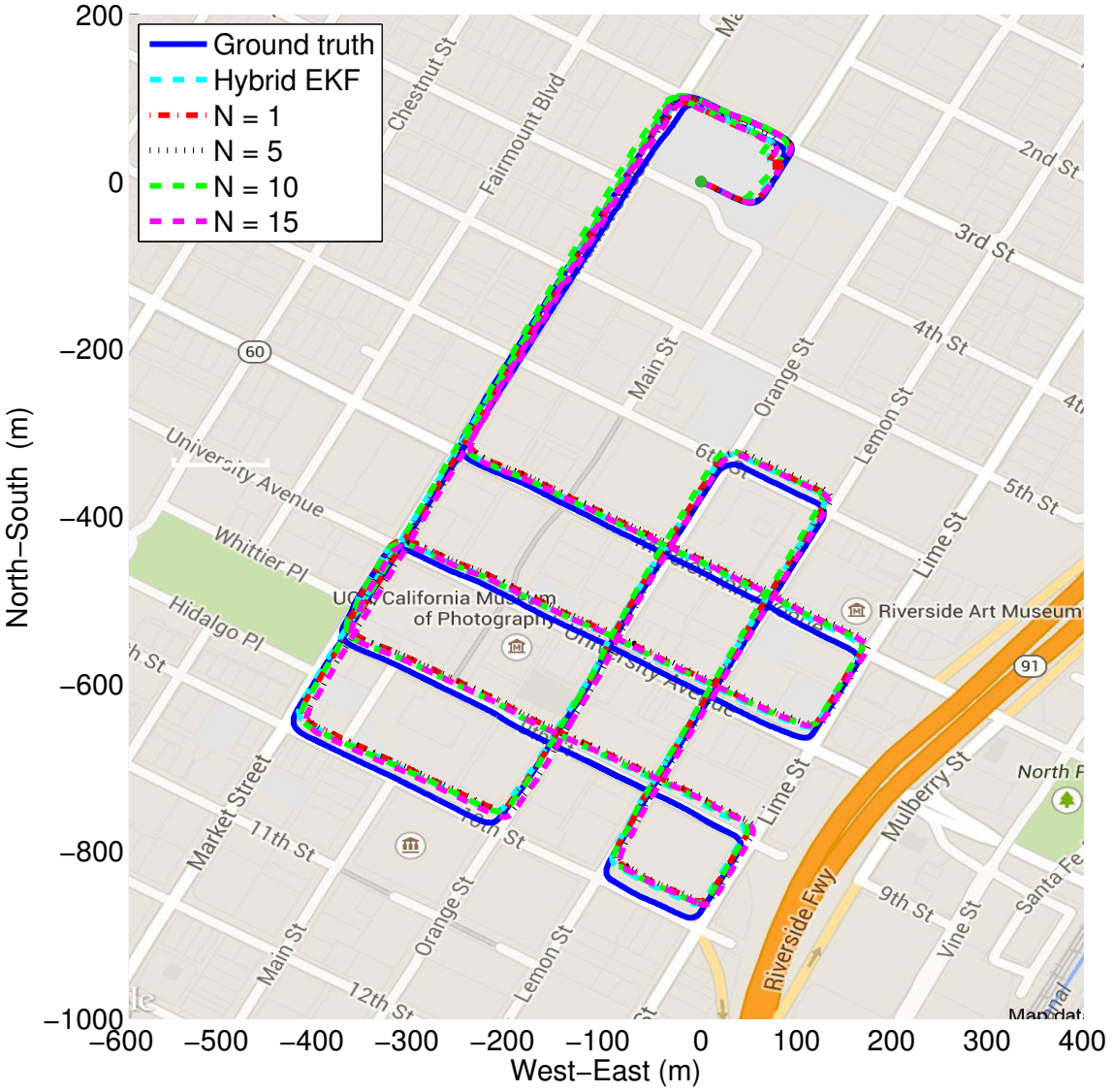

Figure 2.5: Real-world experiment 1: trajectory estimates. The compared methods are the hybrid EKF (dashed cyan line), and the hybrid DEEP-EKF with $N = 1$ (red dash-dotted line), $N = 5$ (black dotted line), $N = 10$ (green dashed line), and $N = 15$ (magenta dashed line). The solid blue line represents the ground truth trajectory.

Figure 2.6: Sample images recorded during the experiment.

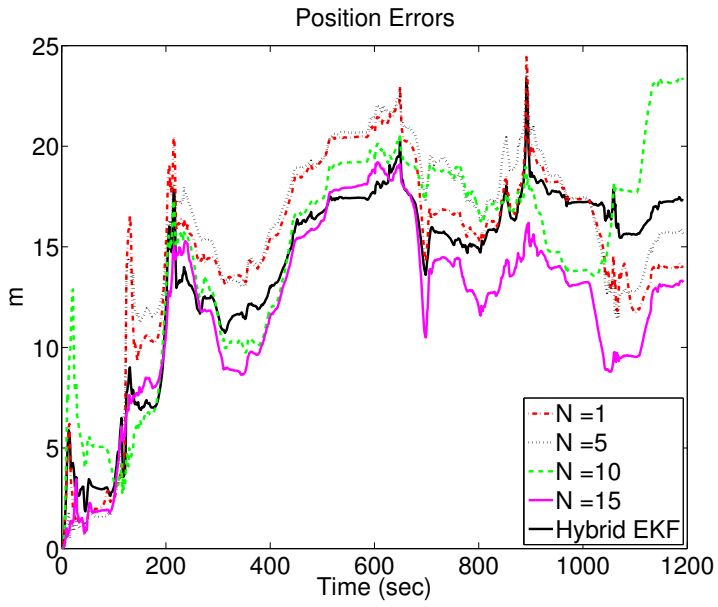

Figure 2.7: Real-world experiment 1: position errors of the hybrid EKF and the hybrid DEEP-EKF for increasing N.

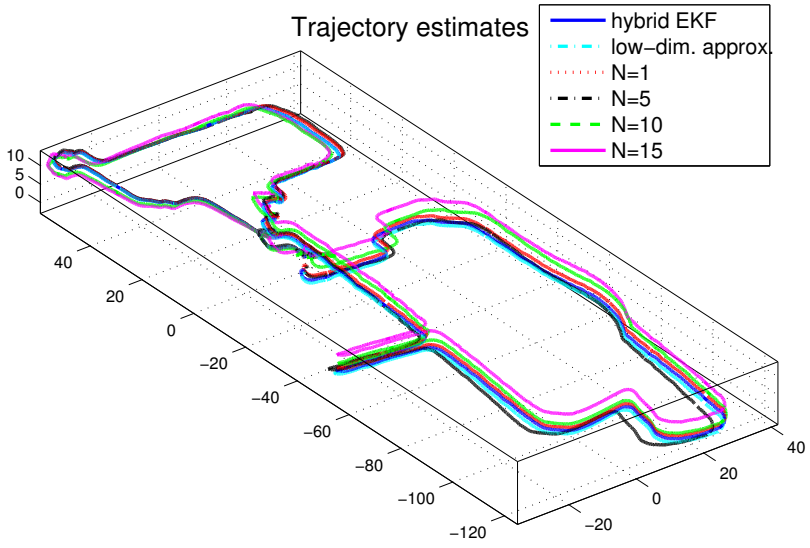

Figure 2.8: Real-world experiment 2: trajectory estimates. The compared methods are: the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$ (blue solid line), $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (cyan dash-dotted line); the hybrid DEEP-EKF with $N = 1$ (red dotted line), $N = 5$ (black dash-dotted line), $N = 10$ (green dashed line), and $N = 15$ (magenta solid line).

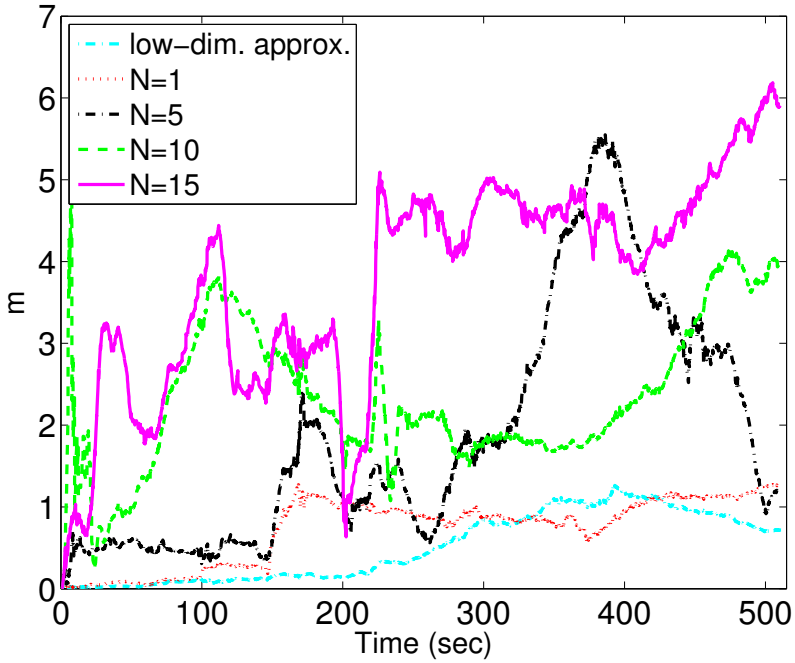

Figure 2.9: Differences of position estimates based on that of the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 1$. The compared methods are: the hybrid EKF with $I_{\mathbf{p}} = I_{\boldsymbol{\theta}} = 0$ (cyan dash-dotted line), the hybrid DEEP-EKF with $N = 1$ (red dotted line), $N = 5$ (black dash-dotted line), $N = 10$ (green dashed line), and $N = 15$ (magenta solid line).

Figure 2.10: Sample images recorded during the experiment.

Chapter 3

High-fidelity Sensor Modeling and Self-Calibration in Vision-aided Inertial Navigation

3.1 Introduction

We now focus on the problem of pose estimation using a low-cost IMU and a rolling-shutter camera. Our interest in the use of a rolling-shutter camera is motivated by the fact that, in most cases, low-cost cameras employ rolling-shutter images, and thus being able to localize with these sensors can lead to the development of widely-available, low-cost systems. Our main focus in this chapter is on the detailed modeling and calibration of the sensors used for pose estimation, a prerequisite for high-quality state estimates. In prior work, the sensors' intrinsic characteristics, relative spatial configuration, and timing, are

calibrated by a combination of offline and online methods. On the one hand, the camera's intrinsic parameters are typically estimated offline via a specialized calibration process. On the other hand, the IMU's biases are typically estimated online, since their values usually drift over time. High-precision estimators typically estimate additional parameters online, such as the camera-to-IMU transformation [41, 48, 64, 103] or the time-offset that exists between the timestamps of the IMU and the camera [59].

However, exiting approaches have a number of shortcomings. First, performing offline camera calibration is an often tedious process, which should be repeated periodically, since mechanical shocks and other factors can lead to changes in the camera parameters over time. Moreover, we note that, even if high-quality offline calibration is performed, the existence of some uncertainty in the camera parameters is inevitable. Treating the computed values as known constants (the standard approach), leads to unmodeled errors, which degrade the accuracy and reliability of any estimator. Finally, in existing approaches some of the non-ideal characteristics of IMUs (e.g., misalignment between the sensor axes, non-unit scale factors) are typically ignored. However, in low-cost MEMS IMUs these effects are likely to be significant, and ignoring them will reduce accuracy.

To address these limitations, in this chapter we present a method for pose estimation and concurrent *self-calibration* that uses high-fidelity models for both the camera and the IMU. Specifically, we are interested in pose estimation in an unknown environment, using inertial measurements and observations of naturally-occurring features tracked by the rolling-shutter camera. We do not use any fiducial points or other knowledge about the

structure of the scene. We demonstrate that, even in this challenging scenario, it is possible to concurrently localize and perform *online* calibration of all of the following quantities:

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

We note that the use of a rolling-shutter camera, as opposed to a global-shutter one, requires special treatment: with a rolling-shutter camera each image row is captured at a slightly different time instant, and therefore, from a different camera pose. Since including in the estimator one state for each image row (the "exact" approach) is computationally intractable, all existing methods employ some assumption about the nature of the camera trajectory (see, e.g., [35,58,79]). 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.

This novel treatment of the rolling-shutter measurements is employed in conjunction with an extended Kalman filter (EKF)-based estimator. This EKF includes in its state vector all the calibration parameters described, in addition to the "standard" states needed for pose estimation (e.g., position, velocity, orientation, and feature positions). Through both Monte-Carlo simulations and real-world experiments we demonstrate that all the calibration parameters can be estimated with high precision. More importantly, however, our results show that jointly estimating the camera motion and *all* the calibration parameters leads to significant improvement in localization accuracy, compared to previous approaches that perform online estimation of only a subset of parameters.

3.2 Related Work

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 [34] for a comprehensive review of calibration methods). Camera self-calibration is also possible. In [11] 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, i.e., the time needed to capture all the rows of the image. 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 [79], or a LED flashing at a known frequency in [72]), a special motion of the camera [46], 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., [41, 48, 73] 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 [8, 92]. 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 [54, 109]. 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 [54] or in an EKF [109], 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.

c) Camera-IMU calibration: The estimation of the *spatial* and the *temporal* relationship between the IMU and camera data streams has been the subject of a number of recent papers. Most work has focused on estimating the transformation (i.e., rotation and translation) between the camera and IMU frames (see, e.g., [41, 48, 64, 103] and references therein), while the temporal calibration between the camera and IMU is a less-explored topic [49,59]. In all cases, however, the intrinsic parameters of the camera are perfectly known, and simplified (bias-only) models for the IMU are used. By contrast, in this work we employ detailed models both the camera and the IMU, whose parameters are estimated online, along with the spatial and temporal relationship between two sensors.

3.3 Estimator formulation

Consider a system equipped with a rolling-shutter camera and an IMU consisting of a 3-axis accelerometer and 3-axis gyroscope. Our goal is to concurrently estimate the state of the system with respect to a global coordinate frame $\{G\}$ and *all* the parameters of the two sensors' models. To track the motion of the system we affix to it a "body" coordinate frame, $\{B\}$, and track the motion of this frame with respect to $\{G\}$. We choose the origin of the body frame as the point where the three accelerometer axes intersect, while its orientation is chosen based on the orientation of the camera frame, $\{C\}$. Specifically, we

define the rotation matrix of $\{B\}$ with respect to $\{C\}$ to be a known, constant matrix[1] ${}^{C}_{B}\mathbf{R}$. The specific value of ${}^{C}_{B}\mathbf{R}$ can be selected at will. For instance, it can be chosen as the identity matrix for simplicity, or set equal to the prior estimate of the camera-to-IMU rotation, so that the axes of $\{B\}$ are "close" to axes of the IMU sensors. In our implementation we use the latter.

3.3.1 Hybrid EKF

The state of the body frame is described by the orientation ${}^{B}_{G}\bar{\mathbf{q}}$, position ${}^{G}\mathbf{p}_{B}$, and velocity ${}^{G}\mathbf{v}_{B}$. The EKF-based estimator that we employ to track this state is a modification of the "hybrid" filter originally proposed in [62]. It combines a sliding-window formulation with a feature-based one, and is able to exploit the computational advantages of both formulations. The resulting hybrid algorithm has a low computational cost, and is capable of real-time operation in devices with limited CPU capabilities, as demonstrated in our experiments. We here briefly describe the structure of the estimator (see Algorithm 1), and refer the reader to [62] for further details. Subsequently, in Sections 3.4 and 3.5 we present the IMU and camera measurement models used in the estimator to enable online sensor calibration.

[1] Defining the orientation of the body frame in this way is necessary because the precise directions of the IMU sensors are not known, and must be estimated. If we chose the orientation of $\{B\}$ independently of the camera, we would need to estimate both the IMU sensors' directions and the orientation of the $\{C\}$ with respect to $\{B\}$. This can be shown to introduce unidentifiable parameters in the system model. Alternative choices do exist, such as aligning one axis of $\{B\}$ with one axis of the IMU [54]. However, that leads to more complex measurement models.

The state vector of the EKF at time-step k is given by[2]:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{\mathbf{c}}^T & \mathbf{x}_{B_1}^T & \cdots & \mathbf{x}_{B_m}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_s^T \end{bmatrix}^T \tag{3.1}$$

where $\mathbf{x}_{E_k}$ is the "evolving state", comprising the current body-frame state as well as the time-varying IMU biases, defined in (3.3); $\mathbf{x}_{\mathbf{c}}$ is the vector of parameters we seek to calibrate, defined in (3.10); the states $\mathbf{x}_{B_j}$, $j = 1 \dots m$ are the body states corresponding to the time instants the past m images were recorded; and $\mathbf{f}_i$, for $i = 1, .., s$ are the currently visible features, in inverse-depth parameterization.

When an IMU measurement is received, it is used to propagate the evolving state and covariance. On the other hand, when a new image is received, the sliding window of states is augmented with a new state. Note that each image is sampled over a time interval of non-zero duration (the rolling shutter readout time). By convention, we here consider that the timestamp associated with each image corresponds to the time instant the *middle* row of the image is captured. Therefore the state corresponding to each image in the filter represents the body-frame state at that time instant.

The images are processed to extract and match point features, which are processed in one of two ways: if a feature's track is lost after m or fewer images, it is used to provide constraints involving the poses of the sliding window and the calibration parameters. For this purpose, the multi-state-constraint method of [64, 74] is employed, which makes it possible to use the feature measurements without including the feature in the EKF state vector. On the other hand, if a feature is still being tracked after m frames, it is initialized

[2]Notation: The preceding superscript for vectors (e.g., G in ${}^G\mathbf{a}$) denotes the frame of reference with respect to which quantities are expressed. ${}^X_Y\mathbf{R}$ is the rotation matrix rotating vectors from frame $\{Y\}$ to $\{X\}$, and ${}^X_Y\bar{\mathbf{q}}$ is the corresponding unit quaternion [99]. ${}^X\mathbf{p}_Y$ represents the origin of frame Y with respect to frame X. $\mathbf{I}$ represents the identity matrix, and $\mathbf{0}$ the zero matrix. Finally, $\hat{a}$ is the estimate of a variable a, and $\tilde{a} \doteq a - \hat{a}$ is the error of the estimate.

in the state vector and any subsequent observations of it are used for updates as in the EKF-SLAM paradigm.

At each time step, the hybrid filter processes a number of features with each of the two approaches. For each feature the appropriate residuals and Jacobian matrices are computed, and a Mahalanobis-distance gating test is performed. All the features that pass the gating test are then employed for an EKF update. At the end of the update, features that are no longer visible and old sliding-window states with no active feature tracks associated with them are removed. Note that, to ensure the correct observability properties of the linearized system model, and thus improve the estimation accuracy and consistency, the hybrid filter employs fixed linearization points for each state [64].

3.4 IMU model and state propagation

Due to the physical characteristics of the sensors, as well as imperfections of the manufacturing process, the IMU measurements are affected by systematic errors (in addition to random noise). For a full description of the sources of errors and their modeling, the reader is referred to [96]. In our work, we employ sensor models that include all the systematic errors that can be modeled linearly and have the most impact on precision.

Let us first consider the accelerometer measurements. Each of the three accelerometer sensors in an IMU provides scalar measurements of specific force, modeled as:

$$a_{m_i} = s_i {}^B\mathbf{u}_i^T {}^B\mathbf{a}_s + b_{a_i} + n_{a_i} \qquad i = 1, 2, 3 \tag{3.2}$$

where ${}^{B}\mathbf{u}_i$ 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 ${}^{B}\mathbf{a}_s$ is the specific-force vector:

$$ {}^{B}\mathbf{a}_s = {}^{B}_{G}\mathbf{R}({}^{G}\mathbf{a}_B - {}^{G}\mathbf{g}) $$

with ${}^{G}\mathbf{a}_B$ being the acceleration of the body frame and ${}^{G}\mathbf{g}$ being the gravity vector. Stacking the measurements of the three accelerometer sensors, we obtain the 3×1 vector:

$$ \mathbf{a}_m = \mathbf{T}_a \, {}^{B}\mathbf{a}_s + \mathbf{b}_a + \mathbf{n}_a $$

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

The gyroscope measurements are modeled as:

$$ \boldsymbol{\omega}_m = \mathbf{T}_g \, {}^{B}\boldsymbol{\omega} + \mathbf{T}_s \, {}^{B}\mathbf{a}_s + \mathbf{b}_g + \mathbf{n}_w $$

where $\mathbf{T}_g$ and $\mathbf{T}_s$ are 3×3 matrices, $\mathbf{b}_g$ is the measurement bias, and $\mathbf{n}_w$ 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 $\{B\}$. 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 [96].

Our goal is to estimate the values of the accelerometer and gyroscope bias, as well as the matrices $\mathbf{T}_a$, $\mathbf{T}_g$, and $\mathbf{T}_s$. We note that by estimating the matrices $\mathbf{T}_a$ and $\mathbf{T}_g$, we are effectively estimating 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 $\{B\}$ is defined by the unit vector corresponding to each row (see (3.2)). Knowing these unit vectors makes it possible to estimate the misalignment of the sensors. Moreover, since the rotation between the camera frame and $\{B\}$ is by definition a known constant, these unit vectors also provide us with the direction of the IMU sensors with respect to $\{C\}$.

Following standard practice, the bias vectors $\mathbf{b}_a$ and $\mathbf{b}_g$ are modeled as Gaussian random-walk processes, while the matrices $\mathbf{T}_a$, $\mathbf{T}_g$, and $\mathbf{T}_s$ are assumed to be uncertain but constant parameters[3]. Therefore, the evolving state of the EKF, $\mathbf{x}_E$, is given by:

$$\mathbf{x}_E = \begin{bmatrix} {}^B_G\bar{\mathbf{q}}^T & {}^G\mathbf{p}_B^T & {}^G\mathbf{v}_B^T & \mathbf{b}_g^T & \mathbf{b}_a^T \end{bmatrix} \tag{3.3}$$

On the other hand, the matrices $\mathbf{T}_a$, $\mathbf{T}_g$, and $\mathbf{T}_s$ are contained in the calibration-parameter vector $\mathbf{x_c}$ (see (3.1)). Specifically, we define a 27×1 vector $\mathbf{x}_{\mathbf{c}_{IMU}}$, comprising all the elements of these three matrices, and this vector is included as part of $\mathbf{x_c}$ (see (3.10)).

IMU-based propagation

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 of the body frame as:

$$\begin{aligned} {}^B\hat{\mathbf{a}}_s &= \hat{\mathbf{T}}_a^{-1}\left(\mathbf{a}_m - \hat{\mathbf{b}}_a\right) \\ {}^B\hat{\boldsymbol{\omega}} &= \hat{\mathbf{T}}_g^{-1}\left(\boldsymbol{\omega}_m - \hat{\mathbf{T}}_s\,{}^B\hat{\mathbf{a}}_s - \hat{\mathbf{b}}_g\right) \end{aligned}$$

[3]Note that, if desired, it is straightforward to also model $\mathbf{T}_a$, $\mathbf{T}_g$, and $\mathbf{T}_s$ by random-walk models. This however was not deemed necessary for the sensors used in our testing.

These are subsequently be used to propagate the estimate of the evolving state via numerical integration, as described in [64]. Moreover, the covariance matrix of the EKF is propagated. For this step, we first compute the Jacobian matrices $\mathbf{\Phi}_k$ and $\mathbf{\Gamma}_k$, which describe the relationship between the evolving-state errors in propagation:

$$\tilde{\mathbf{x}}_{E_{k+1}} = \mathbf{\Phi}_k \tilde{\mathbf{x}}_{E_k} + \mathbf{\Gamma}_k \tilde{\mathbf{x}}_{\mathbf{c}_{IMU}} + \mathbf{w}_k$$

where $\mathbf{w}_k$ is the process-noise error, modeled as zero-mean Gaussian with covariance matrix $\mathbf{Q}_k$. The matrices $\mathbf{\Phi}_k$ and $\mathbf{\Gamma}_k$ are computed analytically, similarly to [64]. Note that the above expression explicitly models the effects that errors in the IMU's parameters have on the state. This is also reflected in the EKF's covariance propagation equation, given by:

$$\mathbf{P}_{k+1} = \begin{bmatrix} \mathbf{\Phi}_k & \mathbf{\Gamma}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{P}_k \begin{bmatrix} \mathbf{\Phi}_k & \mathbf{\Gamma}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \mathbf{Diag}(\mathbf{Q}_k, \mathbf{0}) \tag{3.4}$$

3.5 Camera model

We here present the measurement model of the camera. In doing so, we will introduce all the remaining calibration parameters, which together with $\mathbf{x}_{\mathbf{c}_{IMU}}$ form the parameter vector $\mathbf{x}_\mathbf{c}$ in (3.1). Let us consider that an image with timestamp t is received. Due to the time delays inevitably affecting each sensor, this timestamp is affected by a time offset, t_d, relative to the IMU timestamps [59]. Therefore, the middle image row was actually captured at time $t + t_d$, and an image row that is n rows away from the middle was captured at

$$t_n = t + t_d + \frac{n t_r}{N}$$

where t_r is the image readout time, and N is the total number of image rows. Note that n is a signed quantity: it is positive for rows below the middle, and negative for rows above it.

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:

$$\mathbf{z} = \mathbf{h}({}^C\mathbf{p}_f(t_n)) + \mathbf{n} \tag{3.5}$$

where ${}^C\mathbf{p}_f(t_n)$ 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, which is a modeled as the standard perspective camera with radial and tangential distortion [37]. Denoting ${}^C\mathbf{p}_f(t_n) = [{}^Cx_f \;\; {}^Cy_f \;\; {}^Cz_f]^T$, the image projection is given by:

$$\mathbf{h}\left({}^C\mathbf{p}_f\right) = \mathbf{p}_c + \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix} \begin{bmatrix} u_d \\ v_d \end{bmatrix} \tag{3.6}$$

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

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = d \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2t_1uv + t_2(u^2+v^2+2uv) \\ t_1(u^2+v^2+2uv) + 2t_2uv \end{bmatrix} \tag{3.7}$$

$$d = 1 + k_1(u^2+v^2) + k_2(u^2+v^2)^2 + k_3(u^2+v^2)^3$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^Cz_f} \begin{bmatrix} {}^Cx_f \\ {}^Cy_f \end{bmatrix} \tag{3.8}$$

In the above, k_i, $i = 1, 2, 3$ are the radial distortion coefficients, while t_1, t_2 are the tangential distortion coefficients. Moreover, the position vector ${}^C\mathbf{p}_f(t_n)$ can be written as:

$${}^C\mathbf{p}_f(t_n) = {}^C_B\mathbf{R}\,{}^B_G\mathbf{R}(t_n)\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_B(t_n)\right) + {}^C\mathbf{p}_B \tag{3.9}$$

where ${}^C\mathbf{p}_B$ is the position of the origin of $\{B\}$ in the camera frame, which must be estimated. We can now define the 41×1 vector containing the calibration parameters estimated in the EKF:

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{x}_{\mathbf{c}_{IMU}}^T & {}^C\mathbf{p}_B^T & \mathbf{p}_c^T & a_u & a_v & k_1 & k_2 & k_3 & t_1 & t_2 & t_r & t_d \end{bmatrix}^T \tag{3.10}$$

The estimation of these parameters proceeds as normal in an EKF, by computing the Jacobians of the measurement models with respect to these parameters, and updating their estimates during EKF updates. In this process, the uncertainty of the calibration parameters as well as the effect of this uncertainty on the state estimates, is modeled in a seamless way via the EKF equations.

3.5.1 Rolling-shutter modeling

We now show how the Jacobians of the feature measurements can be computed. It is important to note that, as seen in (3.9), the feature measurements at different rows (i.e., different n) in the *same* image of a rolling-shutter camera depend on states at *different* time instants. Since it is impractical to include in the state vector all these states, previous approaches typically employ assumptions about the nature of the motion during the image readout time (e.g., constant-velocity motion [35, 58], or higher-order parametric forms [36, 79]). However, using low-dimensional models risks losing modeling fidelity, while high-dimensional models incur high computational costs.

We here follow a different approach. Specifically, we begin by observing that in any EKF based estimator (such as the one used here), the processing of the feature measurements requires (i) the residual of each measurement, and (ii) a linear approximation

of the residual describing its dependence on the errors of the EKF state vector. The key idea in our approach is that, in computing the residual, *no* assumptions on the form of the trajectory during the readout time are imposed. Instead, we employ an approximate representation for the *error* during the readout time: we express the error during this time interval as a function of the state error at time the middle row of the image is captured. This, in turn, allows us to only include this one state in the EKF sliding window, thus obtaining a computationally efficient algorithm.

More specifically, the residual corresponding to the feature measurement (3.5) is defined as:

$$\mathbf{r} = \mathbf{z} - \mathbf{h}\left({}^{C}_{B}\mathbf{R}\,{}^{B}_{G}\hat{\mathbf{R}}(\hat{t}_n)\left({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_B(\hat{t}_n)\right) + {}^{C}\hat{\mathbf{p}}_B\right) \tag{3.11}$$

where $\hat{t}_n = t + \hat{t}_d + \frac{n\hat{t}_r}{N}$. To compute $\hat{\mathbf{x}}_B(\hat{t}_n)$, which is needed to evaluate this residual, we integrate the IMU measurements in the time interval $[t + \hat{t}_d, \hat{t}_n]$, starting from $\hat{\mathbf{x}}_B(t + \hat{t}_d)$ (note that backward integration may be necessary). We stress that the state $\mathbf{x}_B(t + t_d)$ is the state at the time the middle row of the image is captured, and is part of the EKF sliding window (see Section 3.3). Thus the estimate $\hat{\mathbf{x}}_B(t + \hat{t}_d)$ is readily available. Note that since we are employing direct integration of the IMU measurements to compute $\hat{\mathbf{x}}_B(\hat{t}_n)$, arbitrary motions can be described (as long as they are within the bandwidth of the IMU).

To derive the linear expression relating the residual to the errors of the state estimates, we begin by directly linearizing the camera observation model in (3.11), to obtain:

$$\mathbf{r} \simeq \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_B(\hat{t}_n) + \mathbf{H}_{\mathbf{p}}\,{}^{G}\tilde{\mathbf{p}}_B(\hat{t}_n) + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}}\tilde{\mathbf{f}} + \mathbf{n} \tag{3.12}$$

where $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{p}}$ are the Jacobians of the measurement function with respect to the orientation and position errors at time $\hat{t}_n$, $\mathbf{H}_{\mathbf{f}}$ is the Jacobian with respect to the feature

position, and $\mathbf{H_c}$ is the Jacobian with respect to $\mathbf{x_c}$. For details on the definition of the orientation error, $\tilde{\boldsymbol{\theta}}_B$, and the computation of the time-related Jacobians, please refer to [59].

We now explain how we express the errors at time $\hat{t}_n$ as a function of the errors at $t + \hat{t}_d$. Starting with the position error, and using Taylor-series expansion, we obtain:

$$^G\tilde{\mathbf{p}}_B(\hat{t}_n) = {}^G\tilde{\mathbf{p}}_B(t+\hat{t}_d) + \frac{n\hat{t}_r}{N}{}^G\tilde{\mathbf{v}}_B(t+\hat{t}_d) + \frac{(n\hat{t}_r)^2}{2N^2}{}^G\tilde{\mathbf{a}}_B + \dots$$

This expression is exact if all terms are kept, but if only a finite number of terms is kept, then we incur a truncation error. The key advantage here is that, since we have prior knowledge about the magnitude of the estimation errors, we can *predict* the worst-case modeling error incurred by any choice of truncation order. For example, if we only keep the first two terms, then the error in each direction is upper bounded by $\frac{(nt_r)^2}{2N^2}\epsilon_a$ where ϵ_a is the worst-case acceleration error. Using $\epsilon_a = 2$ m/sec^2 (which is larger than any value seen in practice in our tests) and $t_r = 33$ msec, the truncation error is upper bounded by 6.9×10^{-5} m along each axis. For the characteristics of the camera used in our experiments, this translates to a worst-case unmodelled reprojection error of 0.08 pixels, when imaging objects at a distance of 2 m.

Thus if we model the position error using two terms, as

$$^G\tilde{\mathbf{p}}_B(\hat{t}_n) \simeq {}^G\tilde{\mathbf{p}}_B(t+\hat{t}_d) + \frac{n\hat{t}_r}{N}{}^G\tilde{\mathbf{v}}_B(t+\hat{t}_d)$$

we only incur a minimal loss of modeling accuracy. A similar analysis for the orientation errors shows that even if we truncate the corresponding series after a single term, thus using the approximation $\tilde{\boldsymbol{\theta}}_B(\hat{t}_n) \simeq \tilde{\boldsymbol{\theta}}_B(t+\hat{t}_d)$, the worst-case unmodelled reprojection errors are below 0.3 pixels, even allowing for rotational velocity errors of 2 deg/sec (which is unrealistically large).

Table 3.1: Simulations: RMS errors of the calibration parameters

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

Using these approximations in (3.12), we obtain the following linearized equation for the measurement residual:

$$\begin{aligned}\mathbf{r} \simeq\ & \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_B(t+\hat{t}_d) + \mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{p}}_B(t+\hat{t}_d) + \frac{n\hat{t}_r}{N}\mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{v}}_B(t+\hat{t}_d) \\ & + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}}\tilde{\mathbf{f}} + \mathbf{n} \end{aligned} \tag{3.13}$$

This expression can be used for EKF updates, as it only involves the errors in the body state at $t+\hat{t}_d$, the feature, and the calibration parameters, which are all part of the EKF state vector.

3.6 Simulations

We present results from two sets of Monte-Carlo simulations demonstrating the performance the proposed algorithm. To obtain realistic simulation setups, we generate the simulated trajectories and the IMU and camera measurements with characteristics (e.g., noise, feature properties) matching those of actual datasets.

First test: For the first 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 collected with the rolling-shutter camera and the Invensense MPU-6050 IMU found on 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.

Table 3.1 shows the RMS errors over 100 Monte-Carlo simulations for all the parameters. The first line corresponds to the initial errors, the second line to the errors after 15 sec of motion, and the third line to the final errors. We note that the initial RMS error values for the $\mathbf{T}_a$ and $\mathbf{T}_g$ 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 3.1) 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. 3.1. These plots show the estimation errors and ± 3 standard deviations reported by the filter for all the calibration parameters. Due to the limited space, for the vector parameters (e.g., biases, the $\mathbf{T}_a$, $\mathbf{T}_g$, $\mathbf{T}_s$ matrices, and so on) we 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 [64]. It is important to note that the results of both Fig. 3.1 and Table 3.1

Table 3.2: Initial standard deviation of the calibration parameters in the second set of simulations

$\mathbf{T}_g$	$\mathbf{T}_a$	$\mathbf{T}_s$ (^{o}/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

suggest that, with sufficiently exciting motion, the calibration parameters are *observable* (identifiable), a result that we seek to prove in future work.

Second test: We now present the results of a second set of simulations, which demonstrates the improvement in performance gained by using high-fidelity IMU models and online calibration. We compare the proposed approach that uses full self-calibration against a "partial calibration" approach, in which only the IMU biases, camera-to-IMU calibration, and the time offset between the camera and IMU is estimated (similarly to [59]). In both cases, the IMU and camera are affected by *small* calibration errors, whose standard deviations are shown in Table 3.2. 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 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

Table 3.3: Second simulation: RMS errors of pose estimation

	${}^{B}_{G}\bar{\mathbf{q}}$ (°)	${}^{G}\mathbf{p}_{B}$ (m)	${}^{G}\mathbf{v}_{B}$ (m/sec)
Full calibration	0.468	0.619	0.027
Partial calibration	2.045	2.914	0.092

zero-mean Gaussian pdfs with the standard deviations given in Table 3.2. Table 3.3 shows the RMS error 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.

3.7 Real-world experiment

We next present results from a real-world experiment, which was conducted using the sensors of a Nexus 4 device. The experiment involves motion in three floors of the UCR

Engineering building, while the device was hand-held. The total duration of the experiment is 9.8 min and the trajectory length is approximately 633 m. The IMU sample rate is 200 Hz, while the images are captured at 22 Hz (sample images are shown in Fig. 3.4). Shi-Tomasi features are extracted in the images [88], and matched by normalized cross-correlation. All data was processed in real-time on the phone, with the average time needed per EKF update being 18 msec. This time is significantly lower than the image period, which shows that the proposed method is capable of real-time operation.

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.

Fig. 3.2 shows the trajectory estimated by the proposed approach, while the reported uncertainties of the IMU orientation and position are shown in Fig. 3.3. 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. Due to limited space, the complete results of the online calibration cannot be included here, but are available online, along with a video of the images recorded during the experiment [60]. 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.

3.8 Conclusion

In this chapter, we have presented an online approach to the joint self-calibration of a rolling-shutter camera and a low-cost IMU during vision-aided inertial navigation. Our approach employs detailed models of the IMU, the camera projection geometry, as well as of the relative spatial configuration and timing of the sensors. Our simulation results and experimental testing demonstrate that 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. These results suggest that by including all the parameters in the state vector, the state remains observable, and the additional parameters are identifiable. A formal proof (or refutation) of this result is the subject of ongoing work.

Algorithm 1 Hybrid EKF algorithm

Propagation: Propagate the state vector using the IMU readings, and the covariance matrix using (3.4).

Update: When camera measurements become available:

- Augment the sliding window with a new state, and begin image processing.
- For each feature track that is complete after m or fewer images, do the following
  - Triangulate the feature using all its observations.
  - Compute the feature-reprojection residuals (3.11) and their Jacobians (3.13), and apply the method of [74] to remove the effect of the feature error.
  - Perform a Mahalanobis-distance gating test.
- For the features included in the state vector, compute the residuals (3.11) and measurement Jacobians (3.13).
- Perform an EKF update using all the features.
- Initialize into the state vector features that are still actively tracked after m images.

State Management:

- Remove SLAM features that are no longer tracked.
- Remove all sliding-window states that have no active feature tracks associated with them.

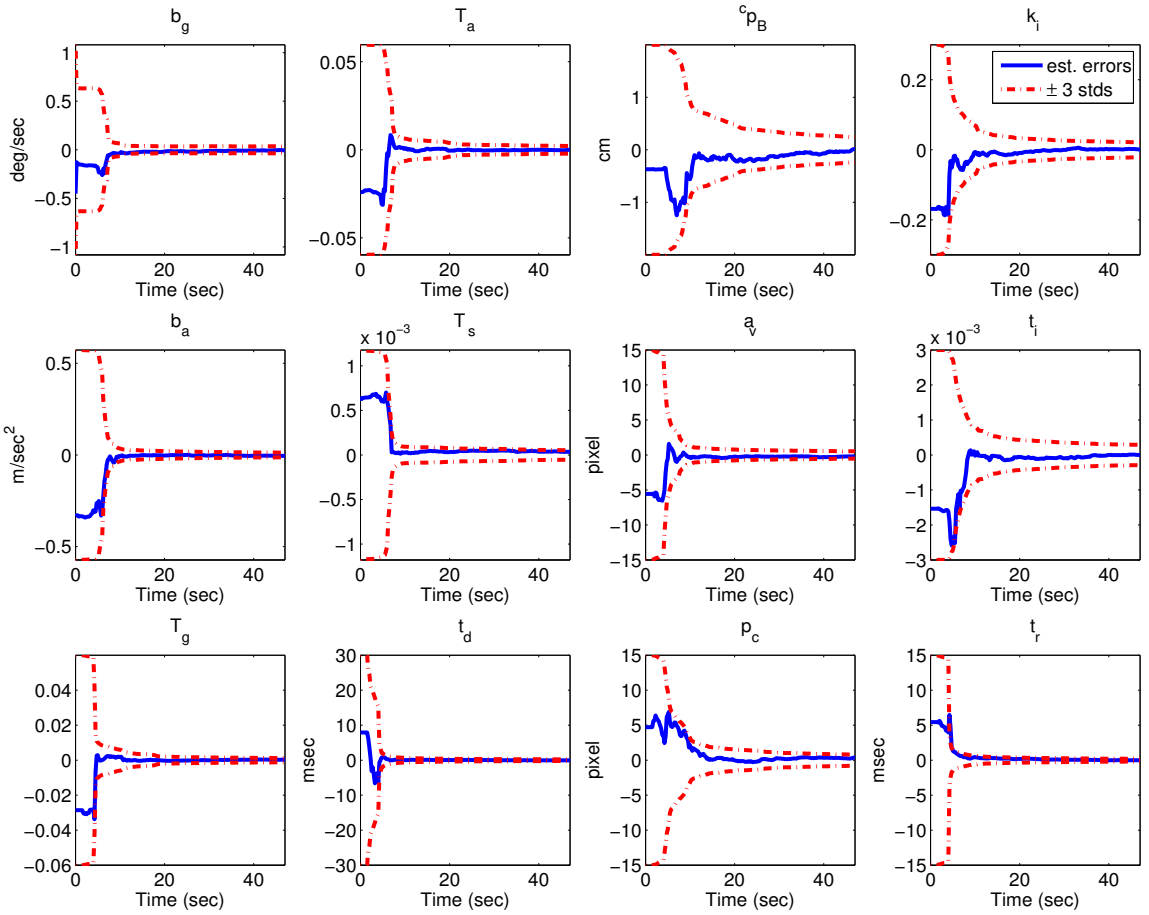

Figure 3.1: Results of a representative simulation trial: estimation errors and the reported ±3 standard deviations. (Top to bottom, left to right) (a) gyroscope biases, (b) accelerometer biases, (c) gyroscope misalignment/scale, (d) accelerometer misalignment/scale, (e) g-sensitivity, (f) camera-to-IMU 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.

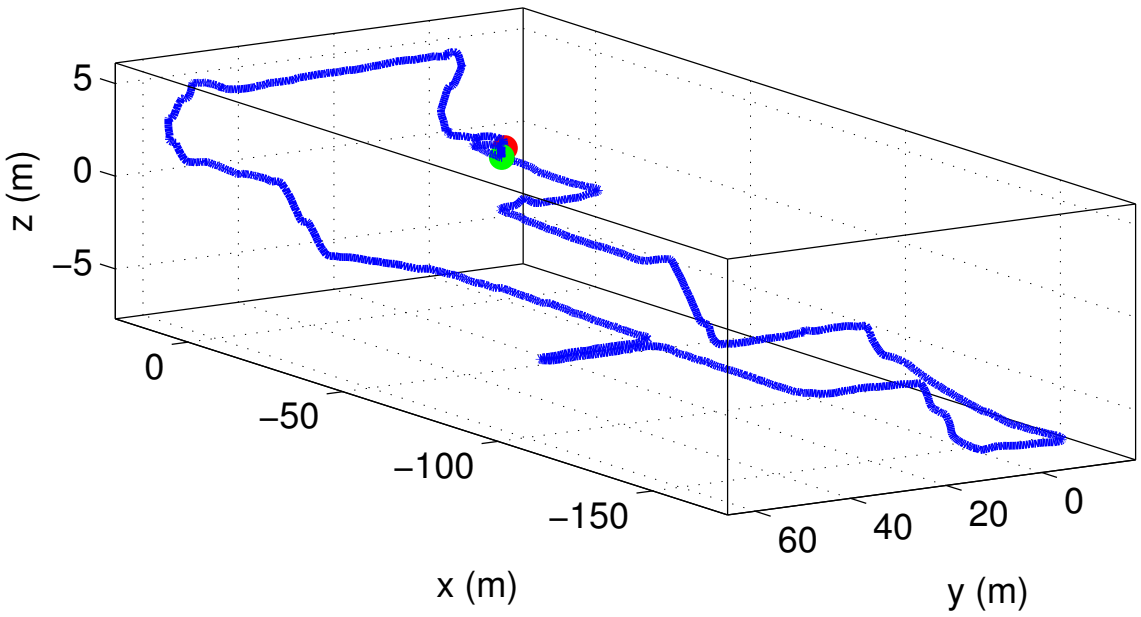

Figure 3.2: Real world experiment: Estimated trajectory. The red mark represents the initial position, while the green mark represents the end position.

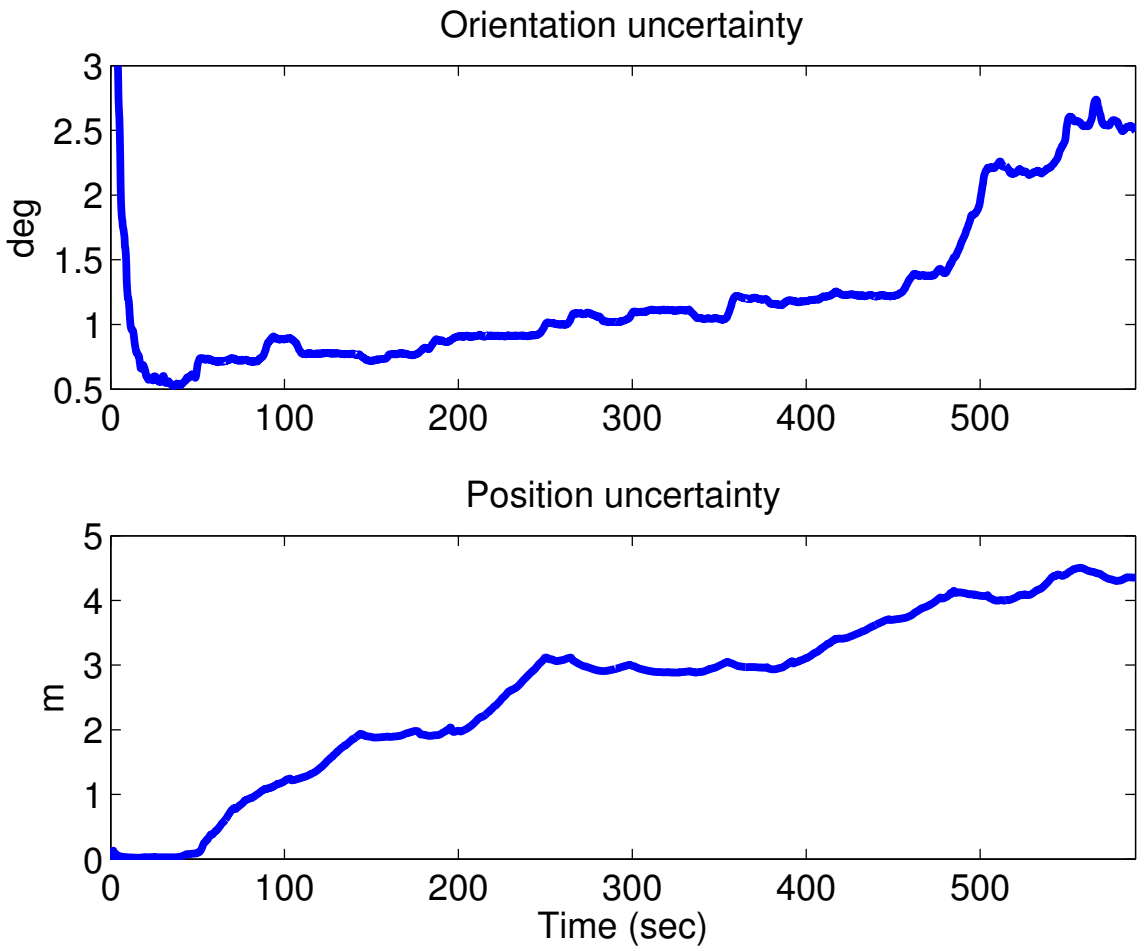

Figure 3.3: Real world experiment: Orientation and position uncertainty (3σ) reported by the EKF during the experiment. The plotted values correspond to the major axis of the uncertainty ellipses. Note the sharp drop in the initial orientation uncertainty, which occurs as the calibration parameters become more certain during the first few seconds.

Figure 3.4: Sample images recorded during the experiment.

Chapter 4

Photometric Patch-based Visual-Inertial Odometry

4.1 Introduction

Among the key challenges that must be addressed in VIO is the fact that cameras naturally produce high-dimensional measurements (e.g., in the order of 10^5 pixels per image). To allow for real-time processing, we must be able to exploit the most useful localization information in the images, while keeping the computational cost low. The "traditional" way of achieving this is by detecting point features in the images (such as SIFT [69], FAST [84], or Shi-Tomasi corners [88]). Typically, only up to a few hundred features are used in each image – a significant reduction in dimensionality compared to the original image size. While the use of point features greatly decreases the number of measurements that need to be processed in the estimator, it also suffers from a number of

drawbacks. First, it results in discarding information from the unused areas in the image. Second, the varying levels of "distinctiveness" of each point feature, which may translate to varying levels of measurement accuracy, are typically not modelled. Third, point-feature extraction may fail altogether in fast-motion or low-light situations, leading to a complete failure of the estimator. Finally, it should be noted that feature extraction and matching may itself be a time-consuming process.

To avoid these shortcomings of feature-based methods, there has been renewed interest in so-called "direct methods", which directly employ the image-intensity measurements in the localization algorithm (see, e.g. [19, 22, 26] and references therein). While, in theory, direct approaches could allow using the measurements of every pixel in an image, and naturally model the local distinctiveness of each image area, they also suffer from shortcomings. The "photometric" (i.e., image-intensity) measurements are sensitive to changes in the camera exposure time and gains, lighting conditions, camera viewing angles, surface properties, and other factors. This can make it difficult to model the relationship between the intensity of the projection of the same scene point in different images. While prior work has offered evidence that direct approaches can lead to improved performance over feature-based ones, the comparisons in the existing literature have generally involved very different systems. This makes it difficult to tease out the effects of using a feature-based vs. a direct approach under the same conditions.

In this work, we describe a new approach for directly using the intensity measurements in distinctive image patches for localization. A key characteristic of the proposed approach is that it models the true irradiance at each pixel as an unknown random vari-

able. Since estimating this random variable is not our primary interest, it is marginalized out during the formulation of the measurement residual. Additionally, in our approach we employ a detailed radiometric camera model that accounts for gamma correction and lens vignetting. In this work, we do not employ the photo-consistency assumption commonly used in direct approaches, and instead model an illumination gain and bias as random variables, to be estimated in our measurement model. Taken together, the above characteristics allow us to accurately model the uncertainty in the image-intensity measurements and their correlation among frames, resulting in increased estimation precision.

The proposed direct measurement model can be applied with several possible estimator formulations (e.g., extended Kalman filter (EKF), sliding window iterative minimization). We here choose to employ this model in conjunction with a sliding-window EKF estimator for VIO, the multi-state-constraint Kalman filter (MSCKF) 2.0 [64, 74]. A key goal of this work is to allow for a direct comparison between the "traditional" point-feature-based approach and the photometric one. To this end, we select the image patches to be used in the photometric formulation around the *same* feature points used in the point-based MSCKF. We perform extensive testing using 50 datasets recorded under varying conditions, each with high-precision ground truth provided by a Vicon system. The results demonstrate that the photometric approach yields, on average, higher localization accuracy, reducing the average position errors by 23%.

4.2 Related Work

Prior work in the area of visual-inertial localization is extensive, and providing a full review within the limited space available is impossible. We here discuss the most relevant approaches, with respect to four different criteria:

Measurement type: Most existing approaches for visual-inertial localization are feature-based ones. The vast majority of these approaches employ point features, but lines have also been used [53, 106]. In contrast to such methods, we here focus on algorithms that directly use image intensities for forming a measurement model. Depending on the type of image regions used in the algorithm, these can be further divided into dense methods, where the entire image is used [77], semi-dense methods, where only regions with large gradient magnitude are used [20, 101], and patch-based methods, where regions around extracted point-features are used [26, 40, 94]. Our approach belongs to the last category.

Camera models: In the feature-based formulation, only a camera's geometric model [16, 37] is generally considered. By contrast, direct approaches also need to model the image formation process, i.e., the mapping from light irradiance to image intensities. A simple, commonly-used model for direct approaches assumes that the measured image intensity at a given pixel is proportional to the irradiance of the incoming light. In practice, however, the camera usually has a nonlinear response function and suffers from lens attenuation. The calibration of these two effects has been considered in [21, 29], and we here also employ a similarly calibrated camera. As shown in [19] (which is a vision-only formulation), modeling these effects can improve estimation performance.

Feature models: For localization in an unknown environment, feature-based approaches generally model the 3D positions of the features as random variables, either to be estimated along with the pose states [41], or to be marginalized to impose constraints on pose states [74, 85], or a combination of both [62]. Similar considerations apply to direct methods. In our approach, feature states are modeled as random variables and marginalized out in the update, thus allowing a probabilistically correct use of the features' information.

In addition to the feature positions, direct approaches must also deal with the *appearance* of the features (or of the collection of pixels considered). Often, this is not modeled in a probabilistic formulation, and instead it is assumed that the appearance (image intensity) of corresponding pixels is the same between images [22, 93]. This assumption, often termed the photo-consistency constraint, is a strong one, and can be violated by changes in the exposure time, scene illumination, or camera viewing angle. A less constraining model is to assume that the *irradiance* is the same between images (the so-called irradiance-consistency assumption). This is done, for instance, in [19]. However, even this approach does not properly model the fact that the actual irradiance is a random variable, that needs to be estimated along with the feature position and possibly other variables. In our work, the irradiance, as well as the illumination gain and bias, are all modeled as random variables in the measurement model.

Estimator choice: Most feature-based visual-inertial algorithms are either Kalman-filter-based methods [41, 48, 74], or methods employing iterative minimization [17, 51, 52]. On the other hand, most direct approaches are formulated as energy minimization problems, and solved by iterative algorithms [14, 101]. Only few EKF-based direct approaches have

been proposed to date [5, 94]. Both of these algorithms employ an EKF state vector that includes the feature positions (in [5] these are represented in a robocentric map), and use the photo-consistency constraint in order to obtain measurement residuals. By contrast, the method we propose here does not include the feature positions in the EKF state vector, which has certain computational advantages, as explained in [74]. Moreover, we formulate a more expressive modified irradiance-consistency constraint, which is able to better model the imaging mechanism.

4.3 Filter Formulation

We now describe the proposed algorithm for visual-inertial localization, which is based on the sliding-window formulation of the MSCKF 2.0 algorithm [64, 74]. Specifically, the state vector of the estimator contains the M poses where the last M images were recorded, while observations of scene features are employed for imposing constraints between these poses. In the original, point-feature-based formulation, these constraints were derived using the image coordinates of the features' projections in the images. By contrast, in the new formulation presented here, the constraints are derived by directly using the image intensity measurements in a patch around each detected feature.

In the remainder of this section we briefly describe the formulation of the state vector, as well as the propagation and state management of the MSCKF algorithm, while the photometric update is described in detail in Section 4.4.

4.3.1 Formulation

We consider a platform equipped with an IMU and a monocular grayscale global-shutter camera, moving in an area populated with naturally-occurring features, whose coordinates are not known a priori. Our goal is to estimate the position and orientation of the platform with respect to a gravity-aligned global coordinate frame, $\{G\}$, using the inertial measurements and the camera images. To derive the estimator's equations, we affix a coordinate frame $\{I\}$ to the IMU, and a coordinate frame $\{C\}$ to the camera. We here assume that the camera is intrinsically calibrated, and the frame transformation between $\{I\}$ and $\{C\}$ is known.

The IMU state at time-step k $\mathbf{x}_{I_k}$ and its error-state $\tilde{\mathbf{x}}_{I_k}$ are defined in (2.16) and (2.17), respectively. The estimator state vector contains the current IMU state, and M states corresponding to the latest M images:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^T & \boldsymbol{\pi}_{k-M}^T & \boldsymbol{\pi}_{k-M+1}^T & \cdots & \boldsymbol{\pi}_{k-1}^T \end{bmatrix}^T \tag{4.1}$$

where each of the states $\boldsymbol{\pi}_\ell$, $\ell = k-M, \ldots, k-1$ consists of the IMU pose at the time the ℓ-th image was recorded, as well as the "illumination parameter" $\boldsymbol{\eta}_\ell$ of the corresponding image (see Section 4.4.4):

$$\boldsymbol{\pi}_\ell = \begin{bmatrix} \mathbf{x}_{\mathbf{p}_\ell}^T & \boldsymbol{\eta}_\ell^T \end{bmatrix}^T, \quad \text{with} \quad \mathbf{x}_{\mathbf{p}_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^T & {}^G\mathbf{p}_\ell^T \end{bmatrix}^T \tag{4.2}$$

4.3.2 Propagation and state augmentation

Every time an IMU measurement is received, it is used to propagate the IMU state and covariance matrix, as described in [64]. Similarly to the original MSCKF, when an image is recorded, a copy of the current IMU pose and the corresponding illumination

parameter are inserted into the state vector (4.1). Specifically, if a new image is recorded at time-step $k+1$, we augment the state vector (4.1) with the IMU-state estimates:

$$\hat{\boldsymbol{\pi}}_{k+1|k} = [\hat{\mathbf{x}}^T_{\mathbf{p}_{k+1|k}} \quad \hat{\boldsymbol{\eta}}^T_{k+1}]^T \tag{4.3}$$

where the initialization of $\boldsymbol{\eta}_{k+1}$ is described in Section 4.4.4. The filter's covariance matrix is also augmented as follows:

$$\mathbf{P}_{k+1|k} \leftarrow \begin{bmatrix} \mathbf{P}_{k+1|k} & \mathbf{P}_{k+1|k}\mathbf{J}_{\mathbf{p}}^T & \mathbf{0} \\ \mathbf{J}_{\mathbf{p}}\mathbf{P}_{k+1|k} & \mathbf{J}_{\mathbf{p}}\mathbf{P}_{k+1|k}\mathbf{J}_{\mathbf{p}}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_{\boldsymbol{\eta}_0} \end{bmatrix} \tag{4.4}$$

where $\mathbf{J}_{\mathbf{p}}$ is the Jacobian of the IMU pose $\mathbf{x}_{\mathbf{p}_{k+1}}$ with respect to the state vector, and $\mathbf{P}_{\boldsymbol{\eta}_0}$ is the initial covariance of the illumination parameter $\boldsymbol{\eta}_{k+1}$. Once the state augmentation is complete, the image is processed so that an EKF update is performed.

4.4 EKF Update

In the MSCKF approach, each feature is being tracked through multiple images. Once the feature is lost, or its track length reaches the length of the sliding window, M, all the feature's observations are used at the same time for an EKF update. We follow the same approach in the proposed photometric formulation of the MSCKF as well. The difference lies in the fact that, while in the original MSCKF the measurement residuals are defined in the space of image coordinates (i.e., the residuals are the feature reprojection errors), in the photometric formulation the residuals are defined in the space of image intensities. Specifically, for each feature we define a planar patch centered around the feature 3D position,

and consider the projection of this patch in each of the images. The measurement residuals are defined by enforcing a modified irradiance-consistency assumption among all images.

In what follows, we describe the geometric model of the camera used in our experiments, our radiometric model that includes the camera response function (gamma correction) and lens vignetting, and finally the formulation of the residuals used for the EKF update.

4.4.1 Geometric model

Consider a point with global 3D position ${}^{G}\mathbf{p}$. The image coordinates of the point's projection in the camera at time step ℓ will be a function of the position vector ${}^{G}\mathbf{p}$, the IMU pose $\mathbf{x}_{\mathbf{p}_\ell}$, and the camera projection geometry. While our approach is applicable with any camera geometry, we here describe the model of [16] that we employ for the fisheye camera used in our experimental setup. With this model, the image projection coordinates are given by:

$$\mathbf{h}\left({}^{G}\mathbf{p}, \mathbf{x}_{\mathbf{p}_\ell}\right) = \frac{1}{r_u \omega} \arctan\left(2 r_u \tan\left(\frac{\omega}{2}\right)\right) \begin{bmatrix} a_u u \\ a_v v \end{bmatrix} + \mathbf{p}_c \tag{4.5}$$

where $\mathbf{p}_c$ is the pixel location of the principal point, (a_u, a_v) are the camera focal length measured in horizontal and vertical pixel units, ω is the distortion parameter, and

$$r_u = \sqrt{u^2 + v^2} \tag{4.6}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^{C_\ell}z} \begin{bmatrix} {}^{C_\ell}x \\ {}^{C_\ell}y \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} {}^{C_\ell}x \\ {}^{C_\ell}y \\ {}^{C_\ell}z \end{bmatrix} = {}^{C}_{I}\mathbf{R}\ {}^{I_\ell}_{G}\mathbf{R}({}^{G}\mathbf{p} - {}^{G}\mathbf{p}_\ell) + {}^{C}\mathbf{p}_I \tag{4.8}$$

In the last equation, ${}^{C}_{I}\mathbf{R}$ is the rotation matrix from the IMU to the camera frame, and ${}^{C}\mathbf{p}_I$ is the position of the origin of $\{I\}$ in the camera frame. We here assume that the camera is both intrinsically and extrinsically calibrated, and therefore the parameters $\mathbf{p}_c, a_u, a_v, \omega$, ${}^{C}_{I}\mathbf{R}$ and ${}^{C}\mathbf{p}_I$ are known.

4.4.2 Radiometric model

In an "ideal" camera, the measured image intensity at a given pixel $\mathbf{p}$ would be proportional to the irradiance of the incoming light at the given pixel:

$$I_{\text{ideal}}(\mathbf{p}) = a\xi(\mathbf{p}) \tag{4.9}$$

where $\xi(\mathbf{p})$ is the light irradiance, and a is a scaling parameter that accounts for the physical size of the pixel on the sensor, as well as the image exposure time. However, in practice, cameras generally have a nonlinear response function to the incoming light energy (the so-called gamma correction), and lenses cause attenuation of the incoming light, which is

typically more pronounced towards the edges of the image (so-called vignetting). Therefore, the measured intensity in a real camera can be modeled as:

$$I_o(\mathbf{p}) = F(V(\mathbf{p})a\xi(\mathbf{p})) + n_p \tag{4.10}$$

where the function $F(\cdot)$ represents the camera response function, $V(\mathbf{p})$ is the lens attenuation at pixel $\mathbf{p}$, and n_p is additive observation noise (e.g., electronic noise).

The camera response function, F, can be estimated via calibration, by taking pictures of a constant scene with several known exposure settings, and creating a lookup table (see, e.g., [21]), or fitting the data with the standard gamma-correction model:

$$F(x) = cx^{\gamma} \tag{4.11}$$

where c and γ are constants to be estimated via fitting. Similarly, the lens attenuation function can be estimated via calibration, e.g., by obtaining images of a uniformly diffused plane. Once these parameters are known, we can obtain the "rectified" image intensity for each pixel in an image:

$$I(\mathbf{p}) = \frac{F^{-1}(I_o(\mathbf{p}))}{V(\mathbf{p})} \tag{4.12}$$

where F^{-1} is the inverse function of F. This rectified intensity value is related to the light irradiance by:

$$I(\mathbf{p}) = a\xi(\mathbf{p}) + n \tag{4.13}$$

where n is additive image-intensity noise. In the remainder of the chapter, the intensity measurements will always refer to the rectified intensity, unless otherwise stated.

In the system used in our experiments, the value of γ is known by design, which removes the need for a calibration of the camera response function. Moreover, we have per-

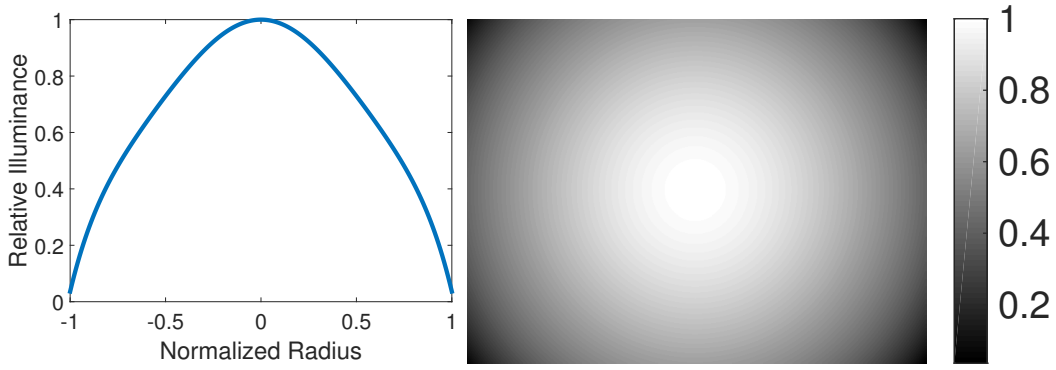

Figure 4.1: Vignetting calibration results. Left: the radially-symmetric vignetting function computed for our camera. The x axis represents the distance from the image center. Right: visualization of the vignetting effect on the image.

formed a calibration of the lens attenuation function using the radially-symmetric vignetting model in [29]. The resulting function V is shown in Fig. 4.1.

4.4.3 Modified irradiance-consistency constraint

We now discuss the formulation of the modified irradiance-consistency constraint, which forms the basis for computing the EKF residuals. Our goal is to derive an equation that relates the observed image intensities at corresponding pixel locations across multiple images. Specifically, let us consider a point feature, f_i, observed in the M images of the sliding window. Our formulation uses the assumption that the scene structure around this feature is *locally* well-modeled by a planar patch P_i. For simplicity, and similarly to [26], in this work we define the normal vector of P_i to be parallel to the optical ray from the feature to the camera, at the time one of the images was recorded (we term this image the

"anchor" image, and select it to be the first image where f_i was seen). The direction of the normal vector is treated as a known constant in our work, but it could also be estimated (and marginalized out later on), if desired.

In our formulation, photometric residuals are computed by considering the image projections of a set of points on P_i. Specifically, we begin by defining N^2 image locations on an $N \times N$ pixel grid centered at the projection of f_i in the anchor image. By "back-projecting" these image coordinates to P_i, we obtain N^2 3D points, with positions ${}^G\mathbf{p}_{i,j}$, $j = 1, \ldots, N^2$. It is important to point out that the selection of the N^2 points in the anchor image is done in a deterministic way, in an $N \times N$ grid around the observed feature coordinates. Therefore, the only random variables that are involved in determining the 3D positions ${}^G\mathbf{p}_{i,j}$ are (i) the IMU pose at the time the anchor image was recorded, $\mathbf{x}_{\mathbf{p}_A}$, and (ii) the distance of P_i to the camera at the time the anchor image was recorded. In our work we parameterize this distance by its inverse, ρ_i. In what follows we use the notation ${}^G\mathbf{p}_{i,j} = {}^G\mathbf{p}(\rho_i, \mathbf{x}_{\mathbf{p}_A}, j)$ to express the fact that ${}^G\mathbf{p}_{i,j}$ is a function of ρ_i, $\mathbf{x}_{\mathbf{p}_A}$, and known quantities.

We now proceed to obtain relationships for the image intensity at the projections of the N^2 patch points on all the M images. From (4.13), we obtain:

$$I_\ell(\mathbf{h}({}^G\mathbf{p}(\rho_i, \mathbf{x}_{\mathbf{p}_A}, j), \mathbf{x}_{\mathbf{p}_\ell})) = a_\ell \xi_{i,j,\ell} + n_{i,j,\ell} \tag{4.14}$$

where $I_\ell(\mathbf{h}({}^G\mathbf{p}(\rho_i, \mathbf{x}_{\mathbf{p}_A}, j), \mathbf{x}_{\mathbf{p}_\ell}))$ is the (rectified) image intensity at the projection of the point ${}^G\mathbf{p}_{i,j}$ in the ℓ-th image, and $\xi_{i,j,\ell}$ is the light irradiance produced by this point in the ℓ-th camera frame. If the surface being imaged was perfectly Lambertian, and the global lighting conditions remained the same, then the irradiance values across all images (i.e., for

all $\ell \in \{k-M, \ldots, k-1\}$), would be the same. In practice, however, this is not the case, and the irradiance may change slightly among images. We model this as:

$$\xi_{i,j,\ell} = \alpha_{i\ell}\xi_{i,j} + \beta_{i\ell} \qquad j = 1, \ldots, N^2 \tag{4.15}$$

In other words, we assume that the irradiance of a patch in different images is related by an (unknown) linear function.

From (4.14) and (4.15), and by combining the measurements from all the N^2 points belonging to the patch, we obtain:

$$\mathbf{I}_\ell(\rho_i, \mathbf{x}_{\mathbf{p}A}, \mathbf{x}_{\mathbf{p}\ell}) = a_{i\ell}\boldsymbol{\xi}_i + b_{i\ell}\mathbf{1} + \mathbf{n}_{i\ell} \tag{4.16}$$

where we have defined $a_{i\ell} = a_\ell\alpha_{i\ell}$ and $b_{i\ell} = a_\ell\beta_{i\ell}$, $\mathbf{1}$ is an $N^2 \times 1$ vector of ones, $\mathbf{I}_\ell(\rho_i, \mathbf{x}_{\mathbf{p}A}, \mathbf{x}_{\mathbf{p}\ell})$ is the vector containing the intensity values at the projections of all N^2 patch points in image ℓ, i.e., a vector with elements:

$$[\mathbf{I}_\ell(\rho_i, \mathbf{x}_{\mathbf{p}A}, \mathbf{x}_{\mathbf{p}\ell})]_j = I_\ell(\mathbf{h}({}^G\mathbf{p}(\rho_i, \mathbf{x}_{\mathbf{p}A}, j), \mathbf{x}_{\mathbf{p}\ell})),\ j = 1, ..., N^2,$$

the vector $\boldsymbol{\xi}_i$ is defined as:

$$\boldsymbol{\xi}_i = \begin{bmatrix} \xi_{i,1} & \xi_{i,2} & \cdot\cdot & \xi_{i,N^2} \end{bmatrix}^T \tag{4.17}$$

and finally $\mathbf{n}_{i\ell}$ is the noise vector, modeled as zero-mean, white, Gaussian, with covariance matrix $\sigma^2\mathbf{I}_{N^2}$.

The equation in (4.16) is the modified irradiance-consistency constraint we employ in this work. It relates the image intensities at the projection coordinates of the N^2 patch points to the irradiance of the patch, as well as the illumination parameters $a_{i\ell}$ and $b_{i\ell}$. Since the projection coordinates are functions of the IMU poses and the patch's inverse

depth, this constraint provides the necessary connection between the geometric and photometric quantities. Note that the patch irradiance and the illumination parameters are unknown random variables, which have to be either marginalized out, or included in the state vector of the filter and estimated. The exact process we follow for this is detailed in the following section, which describes the way the constraint in (4.16) is used for formulating EKF residuals.

4.4.4 Photometric MSCKF update

As explained earlier, at the time when a feature is lost from tracking, or its feature track length reaches the size of the sliding window, all its measurements are used for an EKF update. The process for feature f_i begins by using the feature's projections in the images to obtain an estimate of the feature's inverse depth, ρ_i, via least-squares minimization, as in the original MSCKF algorithm. Using this estimate, as well as the estimate for the IMU poses (available from the MSCKF state vector), we can compute the projection coordinates of the N^2 patch points in all M images, and the image intensities observed at these locations, $\mathbf{I}_\ell(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, \hat{\mathbf{x}}_{\mathbf{p}_\ell})$, $\ell = k-M, \ldots, k-1$. Moreover, an estimate for the "illumination gain" parameter, $a_{i\ell}$, can be obtained as $\hat{a}_{i\ell} = t_\ell / t_o$, where t_ℓ is the exposure time of image ℓ, and t_o is a nominal minimum exposure time. An initial estimate for the "illumination bias" parameter can be chosen simply as $\hat{b}_{i\ell} = 0$. Finally, the irradiance vector $\boldsymbol{\xi}_i$ can be estimated as $\hat{\boldsymbol{\xi}}_i = (\mathbf{I}_A(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, \hat{\mathbf{x}}_{\mathbf{p}_A}) - \hat{b}_{iA})/\hat{a}_{iA}$. Using these estimates, we can compute the following measurement residuals:

$$\mathbf{r}_{i\ell} \doteq \mathbf{I}_\ell(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, \hat{\mathbf{x}}_{\mathbf{p}_\ell}) - \hat{a}_{i\ell}\hat{\boldsymbol{\xi}}_i - \hat{b}_{i\ell}\mathbf{1}, \tag{4.18}$$

for $\ell = k-M, \ldots, k-1$. All these residuals can be stacked in a block vector $\mathbf{r}_i$, whose block elements are $\mathbf{r}_{i\ell}$:

$$\mathbf{r}_i = [\mathbf{r}_{i\,k-M}^T \quad \cdots \quad \mathbf{r}_{i\,k-2}^T \quad \mathbf{r}_{i\,k-1}^T]^T \tag{4.19}$$

This residual vector uses all intensity measurements corresponding to the patch of feature i, across all images. To use it in an EKF update, we must also compute the Jacobian matrices that relate the residual to the estimation errors. To this end, we begin by linearizing (4.18), which yields:

$$\mathbf{R}_{i\ell} \approx \hat{a}_{i\ell}\tilde{\boldsymbol{\xi}}_i + \hat{\boldsymbol{\xi}}_i\tilde{a}_{i\ell} + \tilde{b}_{i\ell}\mathbf{1} - \mathbf{H}_{\rho_{i\ell}}\tilde{\rho}_i - \mathbf{H}_{\mathbf{p}_{i\ell}}\tilde{\mathbf{x}}_{\mathbf{p}_\ell} - \mathbf{H}_{\mathbf{p}_{iA}}\tilde{\mathbf{x}}_{\mathbf{p}_A} + \mathbf{n}_{i\ell} \tag{4.20}$$

where $\mathbf{H}_{\rho_{i\ell}}$, $\mathbf{H}_{\mathbf{p}_{i\ell}}$, and $\mathbf{H}_{\mathbf{p}_{iA}}$ are the Jacobians of the observed image intensities with respect to the feature inverse depth, the IMU pose at time step ℓ and the anchor pose, respectively. Specifically, $\mathbf{H}_{\rho_{i\ell}}$, $\mathbf{H}_{\mathbf{p}_{i\ell}}$, and $\mathbf{H}_{\mathbf{p}_{iA}}$ are block matrices with N^2 rows, with the j-th row given by:

$$\begin{aligned}
\mathbf{H}_{\rho_{i\ell,j}} &= \nabla I_\ell(\mathbf{h}({}^G\mathbf{p}(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, j), \hat{\mathbf{x}}_{\mathbf{p}_\ell}))^T \frac{\partial \mathbf{h}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \rho_i} \\
\mathbf{H}_{\mathbf{p}_{i\ell,j}} &= \nabla I_\ell(\mathbf{h}({}^G\mathbf{p}(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, j), \hat{\mathbf{x}}_{\mathbf{p}_\ell}))^T \frac{\partial \mathbf{h}}{\partial \mathbf{x}_{\mathbf{p}_\ell}} \\
\mathbf{H}_{\mathbf{p}_{iA,j}} &= \nabla I_\ell(\mathbf{h}({}^G\mathbf{p}(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, j), \hat{\mathbf{x}}_{\mathbf{p}_\ell}))^T \frac{\partial \mathbf{h}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \mathbf{x}_{\mathbf{p}_A}}
\end{aligned}$$

where ∇I_ℓ is the image gradient function for image ℓ. We note that the above Jacobians are computed using the first-available estimates of each IMU position, to ensure filter consistency [64]. For the same reason, we do not directly use the image gradient computed from differencing the ℓ-th image in the above calculations. Instead, ∇I_ℓ is computed by transforming ∇I_A to the ℓ-th image.

The residual defined in (4.18) and its linearized approximation in (4.20) involve not only the IMU poses that we are interested in estimating, but also the feature inverse depth, feature-patch irradiance, and the illumination parameters. These are effectively "nuisance parameters," which we can proceed to marginalize, similarly to what is done in the original MSCKF with the feature position. With respect to the illumination gain and bias however, we can explore an additional option: instead of estimating and then marginalizing these parameters "locally," on a per-feature, per-image basis, we can treat one or both of these parameters as being constant for all features within an image. This approach allows us to model "global" effects, such as changes in the entire scene's illumination. To employ this global approach, the parameters can be included in the state vector of the MSCKF, and estimated on a per-image basis (see (4.2)).

In Section 4.5.3 all four options for modeling the illumination gain and bias as either local or global are explored. We here present the case where the illumination bias is modeled as a global parameter, and thus $\boldsymbol{\eta}_\ell = b_{i\ell}$ is included in the state vector (4.1) (and the same value is used for all features in image ℓ), while the illumination gain is modeled as a local parameter. Thus, we can rewrite (4.20) as:

$$\mathbf{R}_{i\ell} \approx \mathbf{H}_{\mathbf{x}_{i\ell}} \tilde{\mathbf{x}}_k + \mathbf{H}_{\mathbf{y}_{i\ell}} \tilde{\mathbf{y}}_i + \mathbf{n}_{i\ell} \tag{4.21}$$

where $\tilde{\mathbf{x}}_k$ is the filter error-state vector at time-step k, $\tilde{\mathbf{y}}_i$ contains the error-states to be marginalized for feature i:

$$\tilde{\mathbf{y}}_i = [\tilde{\boldsymbol{\xi}}_i^T \quad \tilde{a}_{i\,k-M} \quad \cdots \quad \tilde{a}_{i\,k-1} \quad \tilde{\rho}_i]^T$$

and the Jacobians $\mathbf{H}_{\mathbf{x}_{i\ell}}$ and $\mathbf{H}_{\mathbf{y}_{i\ell}}$ are given by:

$$\mathbf{H}_{\mathbf{x}_{i\ell}} = \begin{bmatrix} \mathbf{0} & \cdots & [-\mathbf{H}_{\mathbf{p}_{iA}} & \mathbf{0}] & \cdots & [-\mathbf{H}_{\mathbf{p}_{i\ell}} & \mathbf{1}] & \cdots & \mathbf{0} \end{bmatrix}$$

$$\mathbf{H}_{\mathbf{y}_{i\ell}} = \begin{bmatrix} \hat{a}_{i\ell}\mathbf{I}_{N^2} & \hat{\boldsymbol{\xi}}_i\mathbf{e}_\ell^T & -\mathbf{H}_{\rho_{i\ell}} \end{bmatrix}$$

where $\mathbf{e}_\ell$ is the $(\ell - k + M + 1)$-th canonical basis vector of dimension M. Using (4.21), and stacking these expressions for all the residuals $\mathbf{r}_{i\ell}$, $\ell = k - M, \ldots, k - 1$, we obtain

$$\mathbf{r}_i \approx \mathbf{H}_{\mathbf{x}_i}\tilde{\mathbf{x}}_k + \mathbf{H}_{\mathbf{y}_i}\tilde{\mathbf{y}}_i + \mathbf{n}_i \tag{4.22}$$

where $\mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{H}_{\mathbf{y}_i}$ are matrices with block rows $\mathbf{H}_{\mathbf{x}_{i\ell}}$, and $\mathbf{H}_{\mathbf{y}_{i\ell}}$, respectively, and $\mathbf{n}_i$ a block vector with elements $\mathbf{n}_{i\ell}$.

The expression in (4.22) is the linearized approximation of (4.19) that we sought to obtain. However, since this expression contains both the error of the EKF state vector, $\tilde{\mathbf{x}}_k$, as well as the error vector $\tilde{\mathbf{y}}_i$, which does not involve states in the EKF state, we proceed to compute a new residual that does not include the latter. This is done by a process similar to that used in the original MSCKF. Specifically, we define a matrix $\mathbf{V}_i$ whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{y}_i}$, and define a new residual $\mathbf{r}_i^o$ as:

$$\mathbf{r}_i^o = \mathbf{V}_i^T\mathbf{r}_i \simeq \mathbf{H}_i^o\tilde{\mathbf{x}}_k + \mathbf{n}_i^o \tag{4.23}$$

where $\mathbf{H}_i^o = \mathbf{V}_i^T\mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{n}_i^o = \mathbf{V}_i^T\mathbf{n}_i$. For computational efficiency, we can compute $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ without explicitly computing $\mathbf{V}_i$ [74]. Once $\mathbf{r}_i^o$ and $\mathbf{H}_i^o$ are computed, we proceed by performing a Mahalanobis gating test to reject outliers, and patches whose residuals pass the test are employed in an EKF update, analogously to [74]. Once the update to the state estimate and the covariance matrix are computed, the oldest camera state is removed from the state vector, to maintain a sliding window of bounded length.

4.5 Experimental Validation

In this section we present the results of experimental testing that was carried out to compare the photometric formulation to the original, point-based formulation of the MSCKF, and to examine the effect of a number of parameters on algorithm performance. For this testing, we used a collection of 50 datasets with high-quality ground truth, thus allowing for a thorough performance evaluation. In these experiments, a Project Tango developer tablet was held by a person moving in a room monitored by a Vicon motion-capture system. The duration of each recorded dataset is between 1 and 2 minutes, and sample images from the datasets are shown in Fig. 4.2. A variety of motions were generated, including walking, sudden stopping, running, and fast rotations, to enable evaluation in a wide range of conditions.

During the experiments, an exterior rigid frame with reflective markers was attached to the tablet. The Vicon system provides 500-Hz sub-millimeter accuracy motion-tracking estimates of four markers on the exterior frame. To obtain the transformation between the exterior and the IMU frame, "hand-eye calibration" has been performed offline [13], using the Vicon estimates for the exterior frame and the IMU-trajectory estimates computed via full visual-inertial bundle adjustment. With the calibrated transformation, the position estimates of the markers can then be used to provide the ground-truth estimates for the IMU frame.

In order to isolate the effects of using photometric residuals (as opposed to residuals defined as re-projection errors), in our implementation both the patch-based photometric approach and the original, point-feature MSCKF, employ the same feature tracking and

Figure 4.2: Sample images recorded during the experiments.

matching processes. The same feature tracks are used by all algorithms in the experiments. Moreover, the same point-feature-based triangulation is used to provide the initial guess of the point-feature or patch positions in all cases (note that, if desired, the triangulation could also be formulated based on the photometric residuals).

Since RANSAC is used for outlier rejection in our feature-matching process, different feature tracks will, in general, be generated from the same dataset with different random seeds, and slightly different results will be generated by the filter. Therefore, the result from a single run of an estimator on a given dataset may not be sufficiently representative of the algorithm's performance. To address this issue, we process each dataset with each algorithm 10 times, with a different random seed each time (all the compared algorithms use the same set of random seeds, for a fair comparison). From these results we compute the following performance metrics:

- Typical Error: After processing a dataset with one algorithm, we compute the root-mean-square error (RMSE) of the position estimates over the entire trajectory. Sub-

sequently, we compute the median of these RMSEs over the 10 times the algorithm is run on the same dataset with a different random seed. After repeating this process for all 50 datasets, we compute the average of the 50 results. This metric represents the "expected performance" of the algorithm.

- 90th-percentile error: After processing a dataset with one algorithm, we compute the 90-th percentile of the position error norm throughout the entire dataset. Subsequently, we compute the 90-th percentile of these values over the 10 times the algorithm is run on the same dataset with a different random seed. After repeating this process for all 50 datasets, we compute the average of the 50 results. This metric represents what we expect the (close to) worst-case performance to be.

4.5.1 Patch-based vs. point-feature-based

We first compare the performance of the proposed patch-based formulation against that of the original, point-feature-based MSCKF formulation. For this test, the illumination bias is treated as a "global" parameter in each image, while the illumination gain is treated as a "local" parameter for each feature in each image (as presented in Section 4.4.4). Table 4.1 lists the two performance metrics we are interested in for the two approaches. We can observe that the patch-based approach achieves lower errors than the original point-feature-based approach. Specifically, both the typical error and the 90-th percentile error decrease by approximately 23%. In 66% of the 50 datasets, the patch-based approach achieves errors smaller than the original point-feature-based one. While these reductions may not appear

Table 4.1: Patch-based vs. Point-feature-based Formulation

	Typical Error (m)	90-th Percentile Error (m)
Point-based	0.176	0.270
Patch-based	0.135	0.208

dramatic, they are significant, since the starting point of the comparison (the point-based MSCKF) is already heavily tuned and optimized for accuracy.

These results demonstrate the potential of the direct approach to improve estimation performance. To our knowledge, this is the first time this result has been observed in a setting where the compared algorithms *only* differ in the way in which the residuals are formulated. In previous comparisons that have appeared in the literature, the compared systems typically have significant differences beyond the use of a photometric vs. geometric residual, which makes it difficult to attribute the observed differences in performance to a specific factor.

4.5.2 Effect of camera model fidelity

We now turn our attention to examining the effects of different parameters and design choices on the performance of the direct photometric formulation. In the following experiments, we only employ the patch-based algorithm, but in each experiment we change one aspect of the model while keeping the others fixed, to evaluate the effects of the change.

We start by comparing the effects of using camera models with different levels of detail. In Table 4.2, we present the results of the proposed, "full model," compared to the cases where (i) lens vignetting is not modelled, or (ii) the per-pixel irradiance of the patch is not treated as a random variable, and is instead assumed to be equal to the observed irradiance in the first image. We can see that in both cases, the estimation accuracy is reduced. The loss of performance when vignetting is not modeled is expected, as the camera used exhibits significant vignetting. Prior work has also pointed to the importance of using a detailed model for lens vignetting [29]. However, we note that the need to model the irradiance as an unknown quantity to be estimated is usually ignored in prior photometric approaches. Typically, the irradiance is computed from the intensity measurements at the reference frame, and then treated as a true value (as done for the test in case (ii) here). However, this causes unmodeled errors, because the measurement at the reference frame contains noise that is not accounted for. More importantly, the photometric residuals computed with this irradiance estimate are correlated with each other, but this correlation is not modeled. In our approach, the irradiance is treated as a random variable and thus its estimation error as well as the correlations between the photometric residuals are properly accounted for. This leads to improved performance, as seen in Table 4.2.

4.5.3 Illumination parameters: local vs. global

As discussed in Section 4.4.4, we can either model the illumination parameters (gain and bias) as global ones to be included in the state vector for each image, or as local ones, to be marginalized when processing each feature's measurements. In Table 4.3, we

Table 4.2: Effect of Camera Model Fidelity

	Typical Error (m)	90-th Percentile Error (m)
Full Model	0.135	0.208
No Vignetting	0.139	0.214
No Irradiance	0.148	0.230

Table 4.3: Illumination parameters: Local vs. Global

Illumination Gain	Illumination Bias	Typical Error (m)	90-th Percentile Error (m)
Global	Global	0.151	0.239
Global	Local	0.137	0.212
Local	Global	0.135	0.208
Local	Local	0.149	0.235

present results showing the performance of all four possible combinations. As we can see, while all cases outperform the point-based formulation, the use of a "global" illumination bias and "local" illumination gain outperforms all other options. This indicates that the unmodeled changes in illumination (e.g., due to a flickering light source or exposure-time changes) are better modeled as global parameters for all features in an image, while the effects of non-Lambertian surfaces, which cause changes in irradiance by camera viewpoint, are better modeled as a per-feature, per-image gain.

Table 4.4: Performance with different patch sizes

Patch Size	Typical Error (m)	90-th Percentile Error (m)
4×4	0.137	0.212
5×5	0.135	0.208
6×6	0.139	0.213
7×7	0.150	0.230

4.5.4 Performance with different patch sizes

We evaluated the performance of the algorithm when varying the size of the patches defined around each feature, and the results are shown in Table 4.4. We can observe that the filter's estimation accuracy increases at first, and then decreases as the size of the patch grows. We attribute this to the fact that using a larger patch leads to more information being used (more accurate localization of the features), and this initially leads to improved accuracy. However, when the size of the patch is large, the assumption of the scene being locally planar does not hold anymore, and thus the patch measurements are more likely to be rejected in the Mahalanobis test, resulting in worse performance.

4.6 Conclusion

In this chapter, we have presented a direct formulation of the MSCKF algorithm for VIO. The algorithm utilizes image patches extracted around image feature positions,

and formulates measurement residuals in the image intensity space directly. The proposed method models the true irradiance at each pixel of the patch in the reference image as a random variable, leading to a significant improvement of the estimation accuracy. The formulation of the photometric residual explicitly accounts for the camera response function and lens vignetting (which can be calibrated in advance), as well as unknown illumination gains and biases, which are estimated on a per-feature or per-image basis. Through a detailed experimental evaluation of our algorithm on 50 datasets with high-precision ground truth, we demonstrated that the use of photometric residuals results in increased pose estimation accuracy, with approximately 23% lower estimation errors, on average, in our testing.

Chapter 5

Semi-Dense Visual-Inertial Odometry

In this chapter, we extend the direct VIO algorithm presented in Chapter 4 (henceforth referred to as "direct VIO") to a semi-dense framework where all informative areas in images are used. By doing so, we are able to utilize more information, and thus expect to achieve higher accuracy than both direct VIO and point-feature-based VIO. With the main framework similar to the direct VIO algorithm, semi-dense VIO has two key differences:

- Unlike in direct VIO where triangulation can be performed in geometric space (i.e., by minimizing the feature reprojection errors), in our semi-dense VIO the depth of patches is obtained from photometric triangulation.
- While in direct VIO we define patches around point features, we do not use point features in the semi-dense framework. This can be done by defining informative

patches as the ones with large gradient norm or cornerness score. We choose the later one as it achieves better accuracy in photometric triangulation.

In this chapter, we also propose a novel measurement noise model which accounts for the noise during the image formation, as well as for the interpolation error during the acquisition of intensities at non-integer pixels.

5.1 Photometric Triangulation

Triangulation, the process of determining the position of points (or patches) in 3D space, can be performed either in a geometric or a photometric formulation. In the geometric formulation, image processing is first performed in order to extract and track point features. This information can be utilized to obtain an estimate of the feature depth, by jointly minimizing the reprojection errors. By contrast, with no feature-track information available in the semi-dense method, triangulation is achieved via the minimization of photometric residuals. This photometric triangulation can be formulated in different ways, by selecting different cost functions to be minimized. In this work, we formulate it as minimizing the differences of the normalized intensities between frames, as it better compensates the effects of local illumination changes. More specifically, the patch depth can be estimated as:

$$\rho_i = \arg\min_{\rho_i} \sum_{\ell=1}^{N} \left\| \bar{\mathbf{I}}_\ell(\rho_i) - \bar{\mathbf{I}}_A(\rho_i) \right\|_2^2$$

where ρ_i the depth of patch i, $\bar{\mathbf{I}}_\ell$ the normalized intensity at time-step ℓ, $\bar{\mathbf{I}}_A$ the normalized intensity at the anchor frame where the patch is defined. In order to solve the above non-linear minimization, an initial estimate is obtained by discrete search between the minimal

and maximal allowable depth. A more accurate estimate of the depth is subsequently obtained by polynomial fitting of the cost function around the initial estimate.

A sanity check is performed at the end of the triangulation, in order to validate the correctness of the depth estimate. A "good" depth estimate must meet the following criteria:

- The depth estimate is within the range from the minimal to maximal allowable depth.
- The normalized cross-correlation (NCC) scores between the anchor frame and other frames must *all* be above certain threshold, to ensure the correct matching of patches between frames.
- The resulting intensity residuals computed from the depth estimate in (4.18) must be small. We test this by carrying out a simplified Mahalanobis gating test for the residual. Specifically, we compute:

$$c_i = \sum_{\ell=1}^{N} \mathbf{r}_{i\ell}^T \, \mathbf{R}_{i\ell}^{-1} \, \mathbf{r}_{i\ell}$$

  where $\mathbf{r}_{i\ell}$ and $\mathbf{R}_{i\ell}$ are the residual and covariance of the patch i at time-step ℓ, respectively. c_i is compared against a threshold given by the 95th percentile of the χ^2 distribution with MN degrees of freedom (M is the size of the patch). The test is passed if c_i is smaller than the threshold.

Since the track length of the patch is not available in the semi-dense approach, we start the triangulation with the track length equal to the maximal sliding-window size of the filter. If all the criteria are met, the triangulated patches are used in an EKF update.

Otherwise, we reduce the track length by one and re-triangulate. The above process is repeated until the triangulation is successful or a certain number of trials is reached.

5.2 Intensity Noise Model

We here present a novel noise model for the image intensity measurements. Unlike most localization algorithms where the intensity noise is modeled as identically independent distributed (IID) Gaussian random variables, the proposed noise model takes into consideration various noise sources during the image formation, as well as the error caused by the interpolation when computing intensities at non-integer pixels.

5.2.1 Noise model of the camera

As we have seen in Section 4.4.2, a camera converts the light irradiance, i.e., the photons coming into the sensor per area per second, to the intensity measurement. While an additive IID Gaussian noise is assumed in the intensity measurement model in Section 4.4.2, a typical charge-coupled device (CCD) camera generally is affected by shot noise, thermal noise, quantization noise, and other noise sources, which are generally not IID Gaussian. A simplified noise model is provided in [100], which facilitates the processing of intensity measurements in the filter's framework while maintaining the main characteristics of noises. By applying the noise model of [100] into (4.10), the measured intensity in a CCD camera can be modeled as:

$$I_o(\mathbf{p}) = F(\underbrace{V(\mathbf{p})a\xi(\mathbf{p})}_{I_g(\mathbf{p})} + n_s + n_g) + n_q \tag{5.1}$$

where the function $F(\cdot)$ represents the camera response function, $I_g(\mathbf{p})$ is the noiseless intensity before gamma correction at pixel $\mathbf{p}$, $V(\mathbf{p})$ is the lens attenuation at pixel $\mathbf{p}$, and a is a scaling parameter that accounts for the physical size of the pixel on the sensor, as well as the image exposure time. n_s accounts for the irradiance-dependent noise such as shot noise, n_g accounts for the irradiance-independent noise before gamma correction, and n_q accounts for additional noise such as quantization noise and amplifier noise. n_s, n_g and n_q are modeled as zero-mean, white noise, with variance $I_g(\mathbf{p}){\sigma_s}^2$, ${\sigma_g}^2$ and ${\sigma_q}^2$, respectively.

Following the same procedure as in (4.12), we can compute the "rectified" image intensity for each pixel in an image. By taking its first-order Taylor expansion, the rectified intensity model becomes:

$$I(\mathbf{p}) = \frac{F^{-1}(I_o(\mathbf{p}))}{V(\mathbf{p})} \approx a\xi(\mathbf{p}) + n(\mathbf{p}) \tag{5.2}$$

$$n(\mathbf{p}) = \frac{1}{V(\mathbf{p})}\left(n_s + n_g + \underbrace{\frac{1}{c\gamma} I_g(\mathbf{p})^{1-\gamma}}_{g(I_g(\mathbf{p}))} n_q\right) \tag{5.3}$$

where $n(\mathbf{p})$ is the overall noise in the rectified intensity model, which is modeled as zero-mean, white, Gaussian, with variance equal to $\frac{1}{V(\mathbf{p})^2}\left(I_g(\mathbf{p}){\sigma_s}^2 + {\sigma_g}^2 + g(I_g(\mathbf{p}))^2{\sigma_q}^2\right)$.

5.2.2 Interpolation error

In direct approaches, we generally need to compute image intensities at non-integer image coordinates, a process that requires interpolation of intensities from nearby pixels. Common interpolation algorithms include bilinear, bicubic, and splines (see [83] and references within). Interpolation, which is essentially an approximation of a value from its neighbors, will inevitably cause errors. In what follows, we will derive the upper bound of

errors caused by interpolation, and include it as an additional source of noise into the noise model.

We denote the true intensity and interpolated intensity value at image location $\mathbf{p}$ as $I(\mathbf{p})$ and $\hat{I}(\mathbf{p})$, respectively. The interpolation error is thus defined as:

$$e(\mathbf{p}) = I(\mathbf{p}) - \hat{I}(\mathbf{p})$$

Since no ground-truth value of intensity is available for non-integer coordinates $\mathbf{p}$, there is no way to compute the exact value of the interpolation error. Instead, we can bound the interpolation error. Specifically, as shown in Fig. 5.1, the intensity at non-integer coordinate $\mathbf{p}$ is interpolated from intensities at nearby integer coordinates $\mathbf{p}_{ij}$, $i, j = 0, 1$. We are interested in the maximal error in the interpolation region $\Omega_{\mathbf{p}}$ whose size is 1×1 pixel. We here assume $I(\mathbf{p})$ and $\hat{I}(\mathbf{p})$ are continuous in $\Omega_{\mathbf{p}}$. According to the extreme-value theorem [47], the interpolation error must attain a minimum and a maximum in $\Omega_{\mathbf{p}}$. The extreme values of the error are either obtained at the critical points in the region or along the boundary. In the following, we will consider the maximal possible interpolation error in each case.

Maximal error inside the region

Assume the error $e(\mathbf{p})$ is twice differentiable over the area $\Omega_{\mathbf{p}}$. When there is a critical point $\mathbf{p}_e \in \Omega_{\mathbf{p}}$, $\frac{\partial e}{\partial \mathbf{p}}(\mathbf{p}_e) = \mathbf{0}$. Assume $\mathbf{p}_{ij}$ is the closest integer coordinate near $\mathbf{p}_e$.

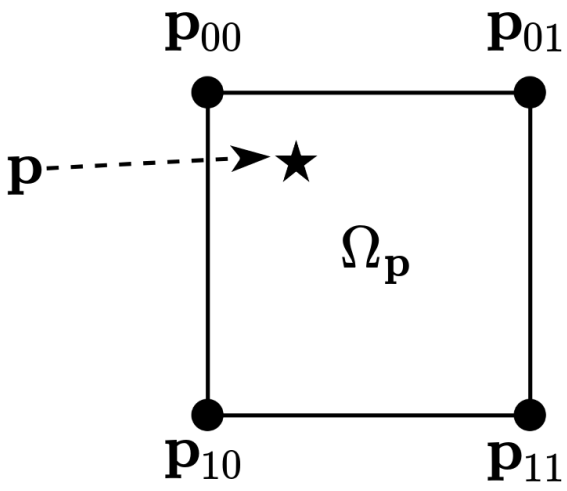

Figure 5.1: An example of interpolation

Note that, by definition, the interpolation error at $\mathbf{p}_{ij}$ is always 0, and by Taylor's theorem, we get:

$$\begin{aligned} 0 = e(\mathbf{p}_{ij}) &= e(\mathbf{p}_e) + \frac{\partial e}{\partial \mathbf{p}}(\mathbf{p}_e)(\mathbf{p}_{ij} - \mathbf{p}_e) + \frac{1}{2}(\mathbf{p}_{ij} - \mathbf{p}_e)^T \nabla^2 e(\boldsymbol{\eta})(\mathbf{p}_{ij} - \mathbf{p}_e) \\ &= e(\mathbf{p}_e) + \frac{1}{2}(\mathbf{p}_{ij} - \mathbf{p}_e)^T \nabla^2 e(\boldsymbol{\eta})(\mathbf{p}_{ij} - \mathbf{p}_e) \end{aligned}$$

Therefore,

$$e(\mathbf{p}_e) = -\frac{1}{2}(\mathbf{p}_{ij} - \mathbf{p}_e)^T \nabla^2 e(\boldsymbol{\eta})(\mathbf{p}_{ij} - \mathbf{p}_e) \tag{5.4}$$

where $\boldsymbol{\eta}$ lies on the line segment joining $\mathbf{p}_e$ and $\mathbf{p}_{ij}$, and $\nabla^2 e(\mathbf{p})$ is the Hessian matrix of $e(\mathbf{p})$.

Define $\boldsymbol{\alpha} = \sqrt{2}(\mathbf{p}_{ij} - \mathbf{p}_e)$. Since $\mathbf{p}_{ij}$ is the closet integer pixel around $\mathbf{p}_e$, $\|\boldsymbol{\alpha}\|_2 \leq 1$. Moreover, since $\mathbf{p}_e$ is assumed to be inside the region, $\|\boldsymbol{\alpha}\|_2 \neq 0$. From (5.4), we get:

$$\begin{aligned} |e(\mathbf{p}_e)| &= \frac{1}{4} \left| \boldsymbol{\alpha}^T \nabla^2 e(\boldsymbol{\eta}) \boldsymbol{\alpha} \right| \\ &\leq \frac{1}{4} | \underbrace{\frac{\boldsymbol{\alpha}^T \nabla^2 e(\boldsymbol{\eta}) \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \boldsymbol{\alpha}}}_{R(\nabla^2 e(\boldsymbol{\eta}), \boldsymbol{\alpha})} | \end{aligned}$$

Since $\nabla^2 e(\boldsymbol{\eta})$ is a Hermitian matrix, according to the min-max theorem [67], $\lambda_1 \leq R(\nabla^2 e(\boldsymbol{\eta}), \boldsymbol{\alpha}) \leq \lambda_2$, where λ_1 and λ_2 are the eigenvalues of $\nabla^2 e(\boldsymbol{\eta})$ in increasing order. Thus,

$$|e(\mathbf{p}_e)| \leq \frac{1}{4} \rho(\nabla^2 e(\boldsymbol{\eta})) \tag{5.5}$$

where $\rho(\mathbf{A})$ is the spectral radius of $\mathbf{A}$.

Note that $\boldsymbol{\eta}$ lies on the line segment joining $\mathbf{p}_e$ and $\mathbf{p}_{ij}$. Since $\mathbf{p}_e$ can be any point in $\Omega_{\mathbf{p}}$, $\boldsymbol{\eta}$ can also be any point in $\Omega_{\mathbf{p}}$. Thus, the interpolation error in the region $\Omega_{\mathbf{p}}$ should be bounded by:

$$|e(\mathbf{p}_e)| \leq \frac{1}{4} \max \left\{ \rho(\nabla^2 e(\boldsymbol{\eta})) \,\middle|\, \boldsymbol{\eta} \in \Omega_{\mathbf{p}} \right\} \tag{5.6}$$

Maximal error on the boundary

The boundary of $\Omega_{\mathbf{p}}$ consists of four sides. On ℓ-th side, assume that two end pixels are $\mathbf{p}_0^\ell$ and $\mathbf{p}_1^\ell$. The interpolation error on ℓ-th side is defined as:

$$e_\ell(x) = I(\mathbf{p}_\ell(x)) - \hat{I}(\mathbf{p}_\ell(x))$$

$$\mathbf{p}_\ell(x) = \mathbf{p}_0^\ell + x(\mathbf{p}_1^\ell - \mathbf{p}_0^\ell), \;\; x \in [0, 1]$$

Since $e_\ell(0) = e_\ell(1) = 0$ by definition and $e_\ell(x)$ is twice differentiable, according to the mean value theorem [86], there must exist at least one stationary point $x_e \in (0, 1)$. Assume that

x_ℓ corresponds to the closet end point near x_e. By following a similar procedure as in the case of the interior of the region, we obtain:

$$e_\ell(x_e) = -\frac{1}{2}(x_\ell - x_e)^2 e''_\ell(\eta) \tag{5.7}$$

where $e''_\ell(\eta)$ is the second derivative of $e_\ell(\eta)$, and η is between x_ℓ and x_e. Since $\|x_\ell - x_e)\| \leq \frac{1}{2}$,

$$|e_\ell(x_e)| \leq \frac{1}{8} \max_{\eta \in [0,1]} \left|e''_\ell(\eta)\right| \tag{5.8}$$

When considering all four sides of the boundary, the interpolation error is bounded by:

$$|e(x_e)| \leq \frac{1}{8} \max_{\ell=1:4} \max_{\eta \in [0,1]} \left|e''_\ell(\eta)\right| \tag{5.9}$$

Computation of the upper bound

The overall upper bound of the interpolation error σ_e is the maximum of (5.6) and (5.9). To obtain numerical values, we note that the error's Hessian matrix $\nabla^2 e(\mathbf{p})$ is equal to:

$$\nabla^2 e(\mathbf{p}) = \nabla^2 I(\mathbf{p}) - \nabla^2 \hat{I}(\mathbf{p}) \tag{5.10}$$

where $\nabla^2 I(\mathbf{p})$ can be computed by convolving given image intensities with selected kernels, and $\nabla^2 \hat{I}(\mathbf{p})$ is determined by the interpolation algorithm. For example, when employing bilinear interpolation,

$$\nabla^2 \hat{I}(\mathbf{p}) = \begin{bmatrix} 0 & \zeta \\ \zeta & 0 \end{bmatrix}$$

$$\zeta = I(\mathbf{p}_{00}) + I(\mathbf{p}_{11}) - I(\mathbf{p}_{01}) - I(\mathbf{p}_{10})$$

The error's second derivative $e''_\ell(x)$ is computed as:

$$e''_\ell(x) = \frac{\partial \mathbf{p}_\ell(x)}{\partial x}^T \nabla^2 e(\mathbf{p}_\ell) \frac{\partial \mathbf{p}_\ell(x)}{\partial x}$$

where $\nabla^2 e(\mathbf{p}_\ell)$ is given in (5.10).

Note that the exact values of $\nabla^2 e(\mathbf{p})$ and $e''_\ell(x)$ can only be computed at the four vertices of the interpolation region. In order to obtain the maximal spectral radius of $\nabla^2 e(\mathbf{p})$ and absolute value of $e''_\ell(x)$ in the interpolation region $\Omega_\mathbf{p}$, the value at the maximum point is required, whose computation would need additional assumptions. For simplicity, we here instead only examine the error bounds at the four vertices and choose the largest one as the upper bound.

The interpolation error is treated as additional noise in the intensity measurement model. Namely, (5.2) becomes:

$$I(\mathbf{p}) = a\xi(\mathbf{p}) + \underbrace{n + e(\mathbf{p})}_{n_e} \tag{5.11}$$

where $e(\mathbf{p})$ is the interpolation error, n_e is the noise including both noise from image formation as well as the interpolation error, and modeled as zero-mean, white, Gaussian, with the covariance $\text{Var}(n_e)=\text{Var}(n)+{\sigma_e}^2$ (σ_e is the upper bound of the interpolation error). The resulting intensity model can be applied to an EKF update with the same procedure as shown in Section 4.4.4, with the only difference being the noise covariance.

5.3 Experimental Results

To test the performance of the proposed approach, we conduct both Monte-Carlo simulations and a set of real-world experiments to evaluate its accuracy and consistency.

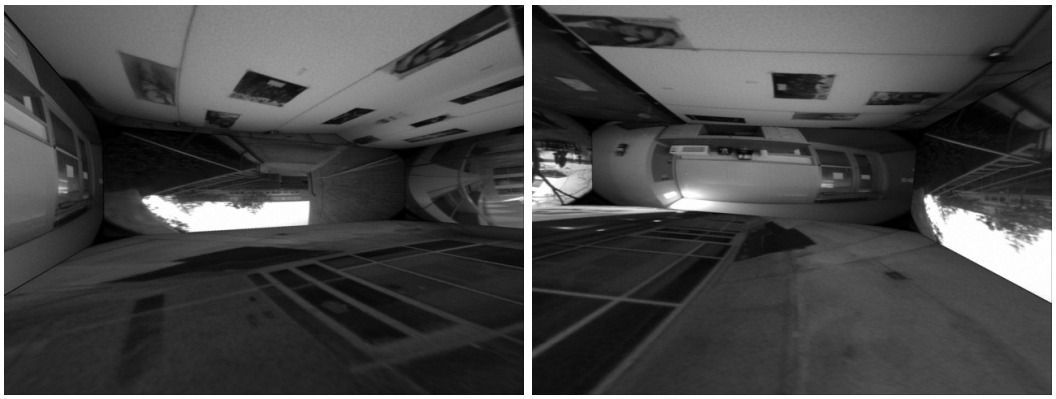

Figure 5.2: Sample images generated in the simulation.

5.3.1 Simulations with Synthetic Images

The ground-truth trajectory in the simulations is generated from a prior experiment, where a hand-held device moves in a room-sized environment. Based on the trajectory, inertial measurements are generated, with different realizations of random noise in each trial. In order to generate synthetic images, we texture-map images that were recorded during a real-world experiment onto the six planes of a rectangular simulated room, including four walls, the floor and ceiling. Shot noise is first applied to the generated synthetic images, modeled as a Poisson random variable. The images are subsequently corrupted with additive Gaussian noise. At the last step, quantization is employed to obtain the 8-bit grayscale images. Note that the noise characteristics in the simulations match what we observe in real-world datasets. Two sample images generated in the simulations are shown in Fig. 5.2. The images are available at 10 Hz, and IMU measurements at 180 Hz. The trajectory is approximately 225 m long, and lasts about 3.27 mins.

Fifty Monte-Carlo simulations are conducted to evaluate the accuracy and the consistency of the proposed methods. The metrics include: i) the RMS error for the IMU orientation and position, and ii) the normalized estimation error squared (NEES) of the IMU pose and velocity at every time step. The RMS error provides a measure of accuracy, while the NEES is a measure of consistency. Specifically, if an estimator is consistent, the average value of the NEES over all Monte-Carlo simulations should equal the dimension of the error-state vector, namely nine in this case.

We begin by examining the effect of using the photometric vs. the geometric triangulation. Fig. 5.3 shows the performance of the point-feature-based approach as well as the direct VIO in different configurations, with the detailed numerical results shown in Table 5.1. As we can see, the direct VIO outperforms the point-feature-based approach in terms of both orientation and position accuracy, which is expected from the results of Section 4.5. Moreover, we note that the use of photometric triangulation in direct VIO results in 20.9% smaller orientation RMSE, and similar position RMSE, compared to that of geometric triangulation, validating the use of photometric triangulation.

We now compare the performance of the semi-dense VIO with different noise models, against that of the direct VIO, shown in Fig. 5.4. From the numerical results in Table 5.2, we can see that all semi-dense approaches outperform the direct VIO. Also, by employing the novel noise model that accounts for noise during the image formation and interpolation error, the semi-dense method achieves 8.6% and 11.8% smaller RMSE in orientation and position, respectively, against the case where the noise is modeled simply as IID Gaussian.

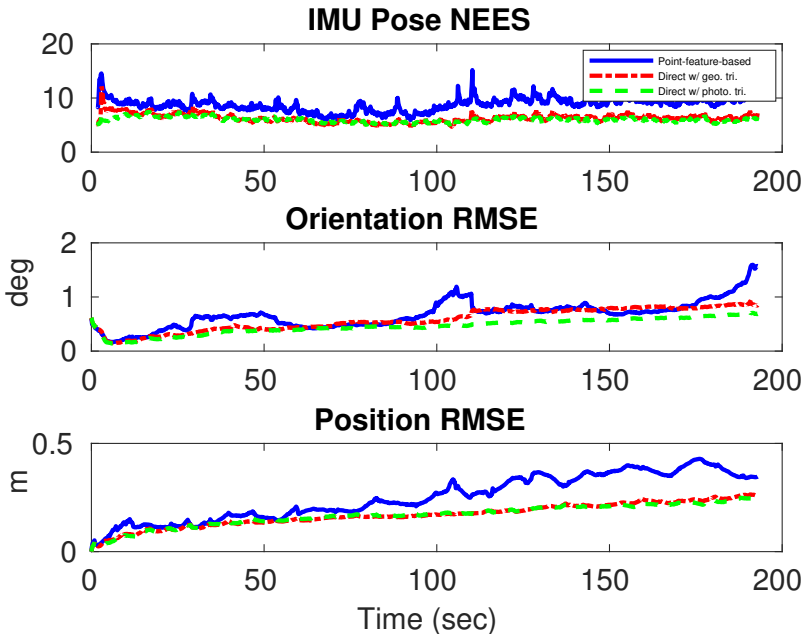

Figure 5.3: Simulation results: the NEES of the IMU pose and velocity, and the position and orientation RMSE over time. Plots are averages over 50 Monte-Carlo trials. The compared methods are: the point-feature-based VIO (blue solid line), and the direct VIO with geometric triangulation (red dash-dotted line), and photometric triangulation (green dashed line).

Note that all direct and semi-dense approaches achieve average NEES smaller that nine, which is caused by the inflation in the standard deviation of noise in the estimator. The inflation is to compensate the unmodeled errors due to: i). the approximation of the shot noise and quantization noise into Gaussian noise, and ii). the assumption that patches are planar and perpendicular to the image ray. We believe that this is one reason why the case which accounts for both noise in image formation and interpolation error is only slightly

Table 5.1: Simulation results: point-feature-based VIO vs. direct VIO with different triangulation

	Point-feature-based VIO	Direct VIO with	
		geometric triangulation	photometric triangulation
Orientation RMSE (deg)	0.671	0.580	0.459
Position RMSE (m)	0.251	0.170	0.168
NEES	8.83	6.31	6.00

better than the case with only modeling noise in image formation, as the interpolation error is already partially compensated by the inflation.

5.3.2 Real-World Experiment

We now present results from real-world experiments, which consist of six datasets in both indoor and outdoor environments. A Tango developer platform was used to record data, with IMU sampling rate at 200 Hz and images at 30 Hz. Two sample images recorded in the experiments are shown in Fig. 5.5. The trajectory lengths in experiments range from 314 m to 465 m. All datasets were processed offline, so that comparisons between the different approaches could be performed.

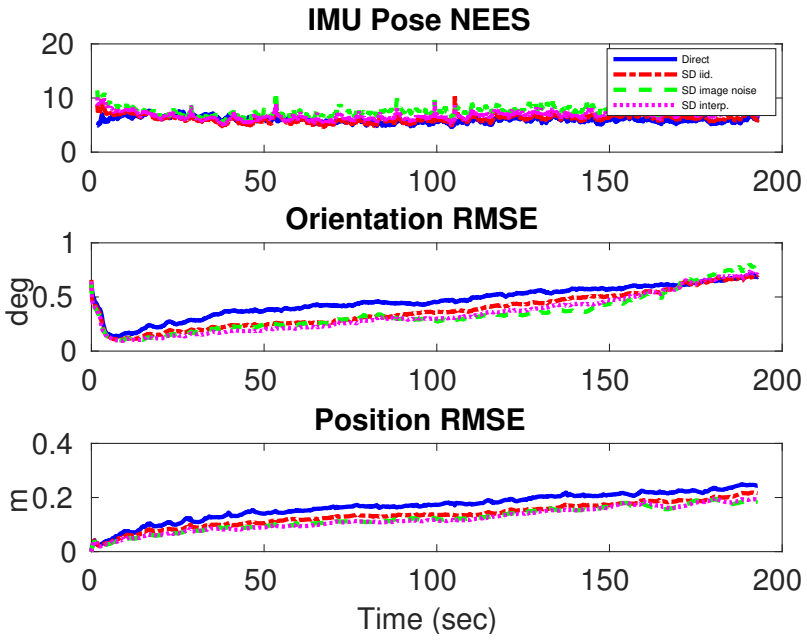

Figure 5.4: Simulation results: the NEES of IMU pose and velocity, as well as RMSE of the position and orientation over time. Plots are averages over 50 Monte-Carlo trials. The compared methods are: the direct VIO with photometric triangulation (blue solid line), and the semi-dense VIO with original IID Gaussian noise (red dash-dotted line), the model accounting for noise in image formation only (green dashed line), and for both noise in image formation and interpolation error (magenta dotted line).

We here evaluate the performance of VIO in different formulations, using the metrics of [28]. Specifically, we compute the position and orientation error as a function of the path length. Since there is no ground truth of the trajectory provided in these mixed indoor-outdoor experiments, we instead employ a global visual-inertial bundle adjustment to obtain the trajectory estimates and treat them as ground truth. Care was taken so that frequent loop-closures occur along the trajectories. This ensures a higher accuracy from the

Table 5.2: Simulation results: direct VIO vs. semi-dense VIO with different noise models

	Direct VIO	Semi-dense VIO, noise model:		
		iid	(5.3)	(5.3) and interpolation
Orientation RMSE (deg)	0.459	0.374	0.344	0.342
Position RMSE (m)	0.168	0.136	0.121	0.120
NEES	6.00	6.19	7.62	6.81

bundle adjustment than other approaches, and thus makes it a reasonable approximation of the ground-truth. For simplicity, we only compare the "best" case in each formulation, namely the point-feature-based VIO, direct VIO with photometric triangulation and semi-dense VIO accounting for both the noise during image formation and interpolation error. One example of the trajectory estimates from these approaches is shown in Fig 5.6.

As defined in [28, 108], to compute the error for a path length L, the errors of the relative-orientation and relative-translation estimates of each method are computed and averaged over the trajectory segments of length L in all datasets. These errors are then

Figure 5.5: Sample images recorded in the experiments.

divided by L, resulting in the position error as a percentage of the distance traveled, and orientation error in degrees per meter traveled:

$$Error_{trans}(L) = \frac{1}{L} average\left(\left\| {}^{I_{t_i}}\mathbf{p}_{I_{t_{i+L}}} - {}^{I_{t_i}}\hat{\mathbf{p}}_{I_{t_{i+L}}} \right\|_2\right)$$
$$Error_{orient}(L) = \frac{1}{L} average\left(\left\| {}^{I_{t_i}}\tilde{\boldsymbol{\theta}}_{I_{t_{i+L}}} \right\|_2\right)$$

where the average is taken over all time intervals $[t_i, t_{i+L}]$ corresponding to trajectory segments of length L, and ${}^{I_{t_i}}\tilde{\boldsymbol{\theta}}_{I_{t_{i+L}}}$ is the error of the relative orientation ${}^{I_{t_i}}_{I_{t_{i+L}}}\mathbf{R}$.

The evaluation of the accuracy of the proposed approaches is shown in Fig. 5.7, which shows the superior performance of the direct VIO and semi-dense VIO over the point-feature-based one. Specifically, when compared with the point-feature-based VIO, the average reduction in error for the direct method, over all path lengths, is 24.8% for orientation and 30.7% for position. In this set of experiments, the semi-dense VIO only achieves a slightly better performance than the direct VIO, which is 28.0% for orientation and 30.9% for position error reduction, compared to the point-feature-based VIO.

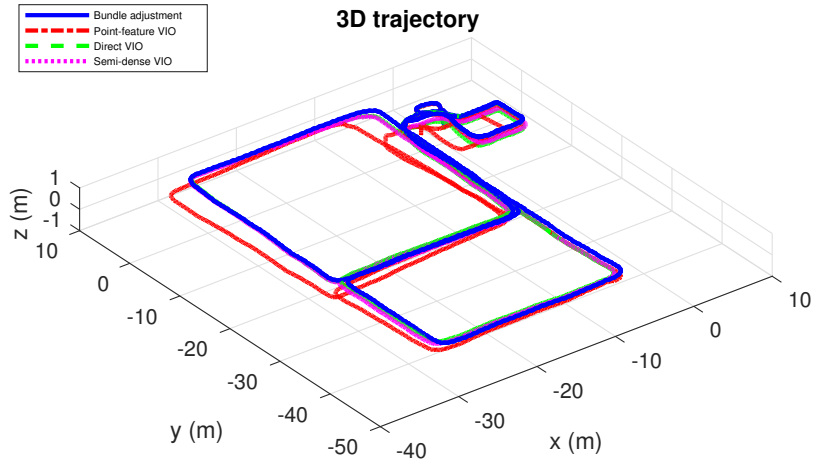

Figure 5.6: Trajectory estimates computed using the point-feature-based (red dash-dotted line), direct (green dashed line), and semi-dense VIO (magenta dotted line), as well as the bundle-adjustment ground truth (blue solid line), in one of the datasets.

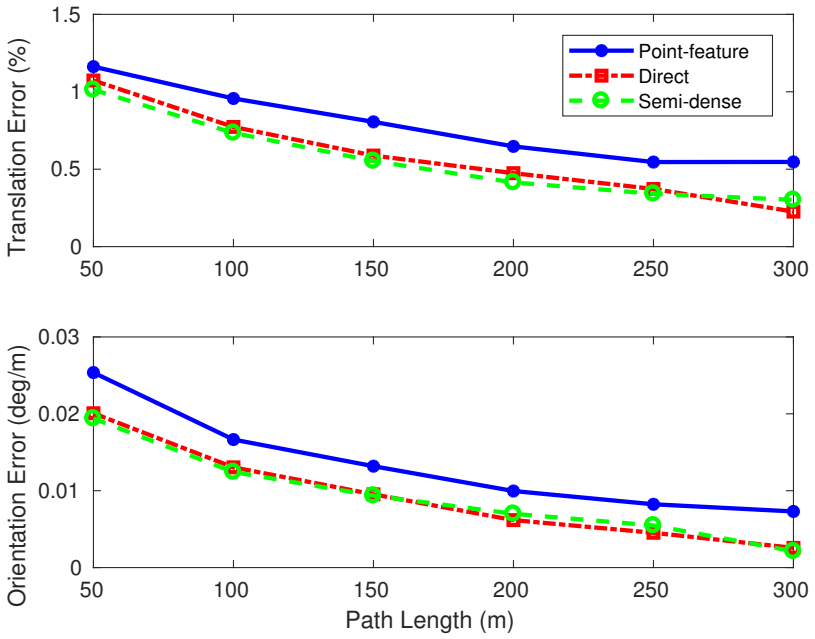

Figure 5.7: Performance comparison between the point-feature-based (blue solid line with points), direct (red dash-dotted line with squares), and semi-dense VIO (green dashed line with circles), by using the estimates of bundle adjustment as ground-truth.

Chapter 6

Summary

Accurate visual-inertial localization is a key technology for many applications such as robotics, virtual reality, augmented reality, and others. Among a number of difficulties in enabling such applications, the main challenges are the requirements of low computational cost and high localization accuracy. In this work, we have proposed the methodologies to address each of the above problems.

Specifically, we first presented the Decoupled Estimate-Error Parameterization (DEEP), which decouples the representation of the trajectory *estimates* from that of the trajectory *errors*. A hybrid EKF-based VIO algorithm is reformulated in the DEEP framework, with enforcing the correct observability properties. The DEEP-EKF formulation is applied to the 3D localization problem using measurements from a rolling-shutter camera, along with *online* self-calibration. Both Monte-Carlo simulations and real-world experiments are conducted to demonstrate that the DEEP formulation yields substantial gains in computational efficiency, while incurring only small loss of accuracy.

To achieve improved estimation accuracy we describe three key methods. First, we propose a high-fidelity sensor modeling, along with *online* self-calibration. The proposed approach is evaluated though Monte-Carlo simulations and real-world experiments, which shows the improved estimation precision compared to existing ones. Second, we presented a direct formulation of the MSCKF VIO algorithm. The algorithm utilizes patches extracted around image point-features, and formulates measurement residuals in the image intensity space directly. A modified irradiance-consistency model is employed, which better in models the illumination changes. A detailed evaluation of the algorithm demonstrates that the use of photometric residuals results in increased pose estimation accuracy compared to geometric ones. Finally, we extend the direct VIO formulation to a semi-dense framework where all informative areas in images are used. Photometric triangulation and a novel noise model, which accounts for noise during the image formation and interpolation errors during acquisition of intensities at non-integer pixels, are employed. Through Monte-Carlo simulations and real-world experiments, we demonstrate the superior estimation accuracy of the proposed semi-dense VIO over the point-feature-based approach and the direct VIO.

Bibliography

[1] M.-C. Amann, T. M. Bosch, M. Lescure, R. A. Myllylae, and M. Rioux. Laser ranging: a critical review of unusual techniques for distance measurement. *Optical Engineering*, 40, Jan 2001.

[2] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

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

[4] Charles Bibby and Ian Reid. A hybrid SLAM representation for dynamic marine environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 257–264, May 2010.

[5] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 298–304, Hamburg, Germany, 2015.

[6] Michael Bosse and Robert 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, AK, May 2009.

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

[8] Pierre-Jean Bristeau, Francois Callou, David Vissire, and Nicolas Petit. The navigation and control technology inside the AR.Drone Micro UAV. In *18th World Congress of the International Federation of Automatic Control*, pages 1477–1484, Milano, Italy, August 2011.

[9] E. Castillo, A. J. Conejo, R. E. Pruneda, and C. Solares. State estimation observability based on the null space of the measurement jacobian matrix. *IEEE Transactions on Power Systems*, 20:1656–1658, 2005.

[10] F. Chenavier and J. L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2588–2593 vol.3, May 1992.

[11] J. Civera, Diana R. Bueno, A.J. 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.

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

[13] Konstantinos Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18(3):286–298, 1999.

[14] A. Delaunoy and M. Pollefeys. Photometric bundle adjustment for dense multi-view 3D modeling. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1486–1493, June 2014.

[15] Frank Dellaert. Square root SAM. In *Proceedings of Robotics: Science and Systems*, Cambridge, MA, June 2005.

[16] Frédéric Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13(1):14–24, 2001.

[17] T.C. 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.

[18] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, June 1987.

[19] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02565*, July 2016.

[20] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[21] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, July 2016.

[22] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[23] D. Farkas, J. Young, B. Baertlein, and U. Ozguner. Forward-looking radar navigation system for 1997 ahs demonstration. In *Proceedings of Conference on Intelligent Transportation Systems*, pages 672–675, Nov 1997.

[24] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2008.

[25] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[26] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, Hong Kong, China, 2014.

[27] Paul Furgale, Timothy D. Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2088–2095, May 2012.

[28] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3354–3361, Washington, DC, USA, 2012. IEEE Computer Society.

[29] Dan B Goldman and Jiun-Hung Chen. Vignette and exposure calibration and compensation. In *The IEEE International Conference on Computer Vision*, pages 899–906, Beijing, China, Oct. 2005.

[30] Jose Guivant and Eduardo 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.

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

[32] Chao Guo, Dimitrios Kottas, Ryan DuToit, Ahmed Ahmed, Ruipeng Li, and Stergios Roumeliotis. Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[33] Seungwoong Gwak, Junggon Kim, and Frank Chongwoo Park. Numerical optimization on the Euclidean group with applications to camera calibration. *IEEE Transactions on Robotics and Automation*, 19(1):65–74, Feb 2003.

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

[35] 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.

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

[37] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, Washington, DC, 1997.

[38] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, January 2013.

[39] Chao Jia and B.L. Evans. Online camera-gyroscope autocalibration for cell phones. *Image Processing, IEEE Transactions on*, 23(12):5070–5081, Dec 2014.

[40] Hailin Jin, Paolo Favaro, and Stefano Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394, 2003.

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

[42] Simon 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 (IROS)*, pages 1251–1256, Maui, HI, Oct. 29-Nov. 3 2001.

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

[44] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365 –1378, Dec. 2008.

[45] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

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

[47] H Jerome Keisler. *Elementary calculus: An infinitesimal approach*. Courier Corporation, 2012.

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

[49] Jonathan Kelly and Gaurav S. 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] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, November 2007.

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

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

[53] D. G. Kottas and S. I. Roumeliotis. Efficient and consistent vision-aided inertial navigation using line observations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1540 – 1547, Karlsruhe, Germany, May 2013.

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

[55] Henrik Kretzschmar, Cyrill Stachniss, and Giorgio Grisetti. Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 865–871, Sept 2011.

[56] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3607–3613, Shanghai, China, May 2011.

[57] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[58] M. Li, B.H. 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.

[59] 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.

[60] Mingyang Li and A. I. Mourikis. Additional results and video. www.ee.ucr.edu/∼mli/ICRA2014.

[61] Mingyang 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.

[62] Mingyang Li and A. I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, Jul. 2012.

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

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

[65] Mingyang Li and Anastasios I. Mourikis. Vision-aided inertial navigation with rolling-shutter cameras. *Int. J. Rob. Res.*, 33(11):1490–1507, September 2014.

[66] Mingyang Li, Hongsheng Yu, Xing Zheng, and A.I. Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 409–416, May 2014.

[67] Elliott H. Lieb and Michael Loss. *Analysis: Second Edition*. American Mathematical Society, 2001.

[68] Steven Lovegrove, Alonso Patron-Perez, and Gabe 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*. BMVA Press, 2013.

[69] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 260(2):91–110, Nov. 2004.

[70] Agostino Martinelli, Viet Nguyen, Nicola Tomatis, and Roland Siegwart. A relative map approach to SLAM based on shift and rotation invariants. *Robotics and Autonomous Systems*, 55(1):50–61, January 2007.

[71] Tad McGeer. Passive dynamic walking.

[72] Marci Meingast, Christopher Geyer, and Shankar Sastry. Geometric models of rolling-shutter cameras. In *Proceedings of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, Beijing, China, 2005.

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

[74] Anastasios I. Mourikis and Stergios 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.

[75] Esha D. Nerurkar and Stergios I. Roumeliotis. Power-SLAM: a linear-complexity, anytime algorithm for SLAM. *International Journal of Robotics Research*, 30(6):772–788, May 2011.

[76] Esha D. Nerurkar, Kejian J. Wu, and Stergios I. Roumeliotis. C-KLAM: Constrained keyframe-based localization and mapping. In *Proceedings of the Workshop on Multi-View Geometry in Robotics, held in conjunction with the Robotics: Science and Systems conference*, Berlin, Germany, June 2013.

[77] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2320–2327, Washington, DC, USA, 2011.

[78] Edwin B. Olson. Real-time correlative scan matching. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4387–4393, May 2009.

[79] Luc Oth, Paul Furgale, Laurent Kneip, and Roland Siegwart. Rolling shutter camera calibration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1360–1367, Portland, OR, June 2013.

[80] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, 113(3):208–219, 2015.

[81] Lina M. Paz, José Guivant, Juan D. Tardós, and José Neira. Divide and conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107 –1120, Oct. 2008.

[82] Pedro Pinies, Todd Lupton, Salah Sukkarieh, and Juan D. Tardós. Inertial aiding of inverse depth SLAM using a monocular camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2797–2802, Roma, Italy, April 2007.

[83] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[84] E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[85] Stergios I. Roumeliotis, Andrew E. Johnson, and James 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.

[86] Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Co., New York, third edition, 1976. International Series in Pure and Applied Mathematics.

[87] David Salomon. *Computer Graphics and Geometric Modeling*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[88] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, Seattle, WA, 1994.

[89] Lingsheng Shi and Nicholas Wormald. Models of random regular graphs. In *In Surveys in combinatorics*, pages 239–298. University Press, 1999.

[90] Malcolm D. Shuster. A survey of attitude representations. *Journal of the Astronautical Sciences*, 41:439–517, October 1993.

[91] Hannes Sommer, Cédric Pradalier, and Paul Furgale. Automatic differentiation on differentiable manifolds as a tool for robotics. In *Proceedings of the International Symposium of Robotics Research*, Singapore, 2013.

[92] Dennis Strelow. *Motion estimation from image and inertial measurements*. PhD thesis, Carnegie Mellon University, November 2004.

[93] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. Real-time dense geometry from a handheld camera. In *Proceedings of the 32nd DAGM conference on Pattern recognition*, pages 11–20, Berlin, Heidelberg, 2010.

[94] P. Tanskanen, T. Naegeli, M. Pollefeys, and O. Hilliges. Semi-direct EKF-based monocular visual-inertial odometry. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6073–6078, Hamburg, Germany, Sept 2015.

[95] Sebastian Thrun, Daphne Koller, Zoubin Ghahramani, Hugh Durrant-Whyte, and Andrew Y. Ng. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7-8):693–716, Aug. 2004.

[96] David Titterton and John Weston, editors. *Strapdown Inertial Navigation Technology*. IEE, 2005.

[97] 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, April 2007.

[98] Nikolas Trawny and Stergios I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. Technical Report 2005-002, Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN, Mar. 2005.

[99] Nikolas Trawny and Stergios I. Roumeliotis. Indirect Kalman filter for 6D pose estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., January 2005.

[100] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of ccd imaging process. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 480–487 vol.1, 2001.

[101] V. Usenko, J. Engel, J. Stckler, and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892, Stockholm, Sweeden, May 2016.

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

[103] 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.

[104] Kejian Wu, Ahmed Ahmed, Georgios Georgiou, and Stergios Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

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

[106] H. Yu and A. I. Mourikis. Vision-aided inertial navigation with line features and a rolling-shutter camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 892–899, Hamburg, Germany, Sept 2015.

[107] Hongsheng Yu and Anastasios Mourikis. Vision-aided inertial navigation with line features and a rolling-shutter camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 892–899, Sept. 2015.

[108] Hongsheng Yu and Anastasios I. Mourikis. Edge-based visual-inertial odometry. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, September 2017.

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

[110] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673 vol.1, 1999.

[111] Sheng Zhao, Yiming Chen, Haiyu Zhang, and Jay A. Farrell. Differential GPS aided inertial navigation: a contemplative realtime approach. In *Proceedings of the 19th International Federation of Automatic Control (IFAC) World Congress*, Cape Town, South Africa, August 2014.

[112] Xing Zheng, Mingyang Li, and A. I. Mourikis. Decoupled representation of the error and trajectory estimates for efficient pose estimation: Supplemental materials. Technical report, University of California, Riverside, 2015.

[113] Xing Zheng, Mingyang Li, and Anastasios Mourikis. Decoupled representation of the error and trajectory estimates for effcient pose estimation. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[114] Xing Zheng, Zack Moratto, Mingyang Li, and Anastasios I. Mourikis. Photometric patch-based visual-inertial odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, May 2017.

[115] Robert Zlot and Michael Bosse. Efficient large-scale three-dimensional mobile mapping for underground mines. *Journal of Field Robotics*, 31(5):758–779, 2014.