Title
Towards Semi-Dense Indirect Visual-Inertial Odometry

Author
Yu, Hongsheng

Publication Date
2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Towards Semi-Dense Indirect Visual-Inertial Odometry

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

Doctor of Philosophy

in

Electrical Engineering

by

Hongsheng Yu

March 2018

Dissertation Committee:

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

The Dissertation of Hongsheng Yu is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Anastasios Mourikis, for his continuous support and mentoring over the past five and half years. His holistic style of rigorous thinking and acute intuition teaches me how to become both a good engineer and research scientist. I would not have come to this far without his lasting support. I would also like to thank other members in my oral and defense committees, Prof. Amit Roy-Chowdhury, Prof. Fabio Pasqualetti, Prof. Jay Farrell and Prof. Qi Zhu, for their valuable suggestions, questions and advices for my research. The discussion helps me a lot during the process.

In addition, I want to thank all the colleagues in my research group. I want to specially thank Dr. Mingyang Li at first. Whenever coding difficulties or theoretical challenges are encountered, he is always willing to help me which is of tremendous value during my early stage of Ph.D life. The knowledge and skills I learned from him have always been beneficial to improve my professional skills. I would also like to thanks Dr. Xing Zheng, Shukui Zhang, the study and research experience with you will be lasting memory during my life. At last, I also want to express my gratitude to all other members at the UCR Control and Robotics Lab, Dr. Dongfang Zheng, Dr. Haiyu Zhang, Dr. Sheng Zhao, Xinyue Kan, Dr. Yiming Chen, Zeyi Jiang for the discussion, work, food and entertainment over the past years. I would also like to thank my friends Bowen Zheng, Dr. Canghong Jin, Fei Ye, Luting Yang, Dr. Peng Wang, Pengxiang Zhu, Qing Zhong, Shan Sun, Shiwen Cheng, Tianshu Wei, Yan Zhu, Yong Ding and Zening Li, for making my Ph.D. life colorful and joyful.

In the end I want to express thanks for all the fund supported for my Ph.D study, without which this work would not be possible. This work was supported by the National Science Foundation (grant no. IIS- 1117957, IIS-1253314 and IIS-1316934), and by Google's project Tango.

Acknowledge of previously published materials: The text of this dissertation, is partly rewritten based on the material that presented in three previously published papers. I am the second author for one paper and first author for the remaining two papers. The papers are as follows.

1. H. Yu, A.I. Mourikis, Edge-based Visual-inertial Odometry, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017

2. H. Yu, 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, 2015

3. M. Li, H. Yu, X. Zheng, 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, 2014

To my parents Elian Deng and Xu Yu for all the support.

ABSTRACT OF THE DISSERTATION

Towards Semi-Dense Indirect Visual-Inertial Odometry

by

Hongsheng Yu

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, March 2018
Dr. Anastasios Mourikis, Chairperson

In this work, we focus on motion estimation in unknown environments using measurements provided by an inertial measurement unit (IMU) and a monocular camera. We are interested in estimating the trajectory of a moving platform, a problem typically termed visual-inertial odometry (VIO). Most of existing methods for vision-aided inertial localization rely on the detection and tracking of point features in the images. These approaches greatly reduce the amount of data to process in each image, and are thus suitable for application in resource-constrained systems. However these treatments inevitably discard information that is beneficial for motion estimation, since not all parts of the images are used. Therefore there has been growing interest in direct methods which rely on directly using image intensities for motion estimation. Although this approach makes it possible to use more pixel locations, it also suffers a number of shortcomings (e.g. non-Lambertian surface properties, and the dependence on camera photometric parameters). By contrast we are interested in approaches that rely on geometry of straight lines or image contours rather than raw image intensities (thus the proposed approaches are indirect methods).

This enables our algorithms to operate in environments where point features are sparse, while circumventing the shortcomings of direct methods.

This thesis is divided into two main parts. We first propose a visual-inertial localization algorithm that employs lines as measurements, in addition to traditional point features. Specifically, a novel parameterization and measurement model for line features are proposed, and we show how line features can be used for self-calibration of the IMU and camera. Our results demonstrate that the proposed approach not only leads to improved localization accuracy in point-feature-poor environments, but also reduces calibration errors compared to the point-only approach.

We then propose a method for monocular visual-inertial odometry that utilizes image edges as measurements. We here relax the requirement for having straight lines, and do not employ any assumption on the geometry of the scene. This enables us to use measurements from all image areas with significant gradient. In addition, we have proposed a novel edge parameterization and measurement model that explicitly account for the fact that edge points can only provide useful information in the direction of the image gradient. Through both Monte-Carlo simulations, as well as results from real-world experiments, we demonstrate that the proposed edge-based approach to visual-inertial odometry is consistent, and outperforms the point-based one.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

Visual-inertial odometry (VIO), which uses the measurements provided by an IMU and a monocular camera, is an attractive solution for localization in GPS-denied environments. The main objective of VIO is to estimate the pose (orientation and position) of a moving platform. Building a feature map of the surrounding environment is not a goal of VIO methods, even though one may be generated as a "by-product" of the estimator. Due to the fact that images provided by visual sensors are high-dimensional measurements, in most existing VIO methods, detection and tracking of point features in the of images are typically employed to reduce the amount of data and maintain a low computational cost. This enables real-time operation even in resource-constraint systems, but inevitably discards information provided by images that is useful for motion estimation. Moreover, the number of point features could be scarce in man-made environments such as indoor office spaces. This may degrade the performance of VIO algorithms if only point features

are used. Therefore there has been growing interest in using more image regions, and direct photometric residuals, for motion estimation. Depending on the nature of the image information used for localization, the approaches could be categorized as dense, where the entire image is used [56], and semi-dense approaches, where regions with significantly large gradient magnitude are used [17, 73].

In this manuscript we are focusing on increasing the density of image regions used for motion estimation in two ways. First, we extract additional types of features in the images: besides extracting commonly-used point features, we also extract and use straight-line features. This is motivated by the fact that in man-made environments straight line features could be prevalent. Therefore by processing them in addition to point features, VIO algorithms could perform better in point-feature-poor environments. Second, we propose using general image contours from the images. This is similar to semi-dense approaches in that all image areas with significant image gradients are used. However our formulation relies on the geometry of contours in the image, instead of raw intensities. The proposed approach provides robustness against scene illumination changes, camera auto-exposure adjustment, and other factors which typically affect the performance of methods that directly use raw image intensities to formulate measurements. Through simulation and real-world experiments, we demonstrate that the proposed edge-based approach can outperform the traditional point-based one. Therefore our proposed approaches are promising components towards the development of a semi-dense VIO system.

1.2 Key contributions

1.2.1 Visual-inertial localization using line features

The first contribution of this work is a visual-inertial localization algorithm that uses line features. This is motivated by the fact that the vast majority of existing methods for vision-aided inertial navigation rely on the detection and tracking of point features in the images. However, in several man-made environments, such as indoor office spaces, *straight line features* are prevalent, while point features may be sparse. Therefore, developing methods that will enable the use of straight-line features for vision-aided inertial navigation can lead to improved performance. While limited prior work on the subject exists, it assumes the use of a global-shutter camera, i.e., a camera in which all image pixels are captured simultaneously. Most low-cost cameras, however, use rolling-shutter (RS) image capture, which renders the existing methods inapplicable. To address these limitations, we here present an algorithm for vision-aided inertial navigation that employs both point and line features, and is capable of operation with RS cameras. The three key contributions of this section are (i) a novel parameterization for 3D lines, which is shown to exhibit better linearity properties than existing ones, (ii) a novel approach for the use of line observations in images, that forgoes line-fitting and does not assume that a straight line in 3D projects to a straight line in the image, and is thus suitable for use with RS cameras, and (iii) a self-calibration approach to calibrate all parameters of a high-fidelity model for both the IMU and camera by using line measurements. Our results demonstrate that the proposed approach not only leads to improved localization accuracy in point-feature-poor environments but also reduces calibration errors compared to a point-only approach.

1.2.2 Edge-based visual-inertial odometry

The second key contribution of our work is a method for monocular visual-inertial odometry that utilizes image edges as measurements. In contrast to previous feature-based approaches, the proposed method does not employ any assumption on the geometry of the scene (e.g., it does not assume straight lines). It can thus use measurements from all image areas with significant gradient, similarly to direct semi-dense methods. However, in contrast to direct semi-dense approaches, the proposed method's measurement model is invariant to linear changes in the image intensity. In addition we propose a novel edge parameterization and measurement model that explicitly account for the fact that edge points can only provide useful information in the direction of the image gradient. We present both Monte-Carlo simulations, as well as results from real-world experimental testing, which demonstrate that the proposed edge-based approach to visual-inertial odometry is consistent, and outperforms the point-based one.

1.2.3 Manuscript organization

This manuscript is organized as follows. In Chapter 2 we focus on the problem visual-inertial localization by incorporation of line features. We show that improved performance can be attained by using line features and how lines can be used for sensor self-calibration. In Chapter 3 we describe the edge-based visual-inertial odometry. Through Monte-Carlo simulations and real-world experiments, we demonstrate that the proposed approach out-performs an analogous feature-based approach.

Chapter 2

Visual-Inertial Localization with Line Features

2.1 Introduction

In this chapter, we focus on the problem of motion estimation using inertial measurements and visual observations of line features with a rolling-shutter (RS) camera. A significant body of literature has focused on the problem of motion estimation using cameras and inertial measurement units (IMUs). However, prior work typically assumes that the camera has a global shutter (GS), i.e., that all the pixels in an image are captured simultaneously. By contrast, most low-cost cameras typically use CMOS sensors with a RS, capturing each row of pixels at a slightly different time instant. To develop high-precision localization methods, suitable for low-cost robots and for indoor navigation using consumer-grade cameras, the RS effect must be explicitly addressed in the design of estimation algorithms.

The (only few) methods that have been proposed to date for motion estimation using an IMU and a RS camera rely on the detection and tracking of point features in the scene. Point features are commonly used in motion-estimation methods, as they are abundant in many real-world environments, and well-established algorithms exist for their detection and tracking. In several man-made environments however, such as indoors and in urban areas, *straight-line* features are equally common. In this work, we present an algorithm that is able to use line features for motion estimation with a RS camera, either as an alternative to or in addition to point features.

As discussed in Section 3.2, the subject of motion and structure estimation from line features has been extensively studied. However, almost all prior work assumes that a GS camera is used. This is a significant assumption, since it guarantees that straight lines in the scene project into straight lines in the image. Consequently, the first step of all methods for line-based motion estimation using GS cameras is to perform straight-line detection and line-fitting, in order to obtain the equations of image lines. When a RS camera is used, straight lines do *not* in general project into straight lines in the images, and therefore these methods are not applicable. If restrictive assumptions on the camera motion (e.g., constant velocity) are imposed, it is possible to obtain a parametric description of the projection of a straight line (see, e.g., [1]). However, such assumptions do not hold in general-motion cases, and thus imposing them would lead to loss of estimation precision.

The main contribution of this chapter is a formulation of line-based visual-inertial motion estimation with a RS camera that addresses the above challenges. Specifically, we develop a measurement model for straight lines that is based on using the observations of

all pixels on the projection of a line individually. This approach makes a line-fitting step to compute the equations of image lines unnecessary, and can thus be employed with either a RS or a global-shutter camera. This measurement model is used in conjunction with a novel minimal parameterization for straight lines. We demonstrate experimentally that this parameterization exhibits better linearity than parameterizations proposed in prior work, and is thus better suited for use in linearization-based estimators.

The new formulation of the line-measurement equations is general, and can be employed for estimation in a variety of settings (e.g., using either known or unknown lines), with either an extended Kalman filter (EKF) or an iterative-minimization method. We here assume that the positions and directions of the lines in the scene are not known *a priori*, and present a visual-inertial odometry algorithm based on the hybrid-EKF algorithm of [47]. In order to enable the use of a RS camera, we employ the formulation of [45], which makes it possible to use the RS-camera measurements *without* imposing any assumptions on the camera motion.

We also note that to obtain high-precision state estimates in vision-aided inertial navigation, one pre-requisite is the use of accurate sensor models for both the camera and the IMU. The additional information provided by line features can be employed to improve the accuracy of both localization and sensor calibration. Therefore, in this chapter we also propose a self-calibration framework that uses both point and line measurements, to concurrently localize and estimate the following parameters:

- 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

Our simulation and experimental results demonstrate that the proposed method yields high-precision state estimates, and high-quality calibration of the parameters in sensor models.

2.2 Related Work

We begin by discussing related work, divided into four areas:

a) Motion Estimation with a Rolling-Shutter Camera: The use of a RS camera, as opposed to a GS one, for motion estimation requires a special treatment: with a RS camera each image row is captured at a slightly different time instant, and therefore from a different camera pose. Since including one state for each image row (the "exact" approach) is computationally intractable, existing methods employ some assumption about the nature of the camera trajectory [26, 42, 58]. 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 *er-*

rors during the image readout time. Since these errors are typically small, this leads to only small modeling inaccuracies.

b) Structure from Motion and SLAM using Line Features: In both the computer vision and robotics communities, several approaches have been proposed to estimate the motion of camera and scene structure by extracting line features from images (see, e.g., [5, 9, 16, 70, 76] and references therein). However these approaches all employ GS cameras, and thus suffer from the limitations discussed in Section 2.1. Similarly, the work of [39], which employs line-feature observations in conjunction with inertial measurements, also utilizes a GS camera. By contrast, motion estimation using straight-line features and RS camera is a less-explored topic. In [1] a bundle-adjustment method that employs constant-velocity motion assumptions is employed, while in [16] prior information about the lines' directions in space is assumed. By contrast, in our work none of these assumptions are necessary.

c) Line Measurement Models and Line Parameterizations: When a GS camera is used, the projection of a 3D straight line in an image is a straight line. This property, which is employed in all prior work on motion estimation with lines and a GS camera, is no longer valid when a RS camera is used. To address this issue, we here propose a new way of using the "raw" pixel measurements belonging to line features, which is applicable with both types of cameras.

In addition to the way in which line measurements are processed, the parameterization of a line will also have great impact on the performance of any linearization-based estimator. Previous work has used Plücker coordinates [5] or a pair of vectors [13, 65] to represent 3D lines in space. However, these parameterizations are not minimal, and

this over-parameterization can lead to numerical issues in EKF-based estimators. We here propose a novel, minimal error parameterization for 3D lines, which has favorable characteristics. Specifically, it has inverse-depth properties, and is anchored in one of the camera poses from which the line was observed. These properties are desirable in EKF-SLAM, as discussed in [66]. Moreover, we demonstrate that the proposed parameterization has favorable linearity characteristics, through an analysis similar to that in [66].

d) Calibration of visual and inertial sensors: In [46] sensor models that include all the systematic errors that can be modeled linearly and significantly affect the precision are presented. In addition an EKF-based self-calibration approach is presented to calibrate all the parameters. However, this approach is not directly applicable for line measurements when only coarse estimates of the calibration parameters and the initial state are available, due to the increased nonlinearity inherent in the line measurement model. Therefore in this paper we implement a nonlinear-minimization based formulation that can process the line measurements to calibrate all parameters described earlier, as well as initialize the trajectory estimates. After convergence, the line measurements can be used similarly to point feature for sensor self-calibration as proposed in [46].

2.3 Line Parameterization

We begin by discussing the line parameterization we employ in our work, which we term the two-point inverse depth parameterization (TPIDP). For each line we employ an "anchor frame" $\{A\}$, which is a known, constant coordinate frame (e.g., the first frame from which the line was observed), and use it to define the quantities used in the parameterization.

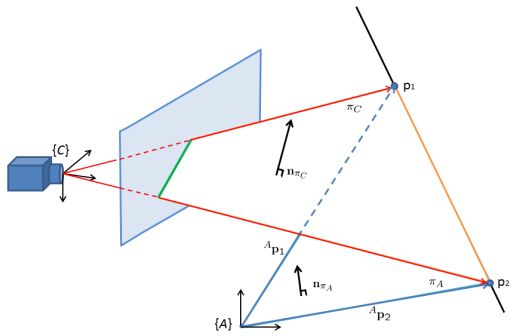

Figure 2.1: Illustration of the line-projection geometry. The origin of the anchor frame A and the line define a plane with normal vector $\mathbf{n}_{\pi_A}$, while the origin of the camera frame $\{C\}$ and the line define a plane with normal vector $\mathbf{n}_{\pi_C}$.

Let the position and orientation of the anchor frame with respect to the global reference frame, $\{G\}$, be denoted as ${}^G\mathbf{p}_A$ and ${}^G_A\mathbf{R}$, respectively[1]. To present our proposed line parameterization, we note that when a line is observed from a camera frame $\{C\}$, the normal vector of the plane defined by the line and the camera optical center, $\mathbf{n}_{\pi_C}$, is the only quantity needed in order to derive the measurement equations (see Section 2.6.2). To describe the line parameters that we define, we proceed to obtain an expression for $\mathbf{n}_{\pi_C}$.

We start by considering two distinct points, $\mathbf{p}_1$ and $\mathbf{p}_2$ on the line. The unit vector normal to the plane defined by the origin of the camera frame and the line, expressed with respect to the camera frame, can be computed as:

$$ {}^C\mathbf{n}_{\pi_C} \sim {}^C\mathbf{p}_1 \times {}^C\mathbf{p}_2 $$

where $\sim$ denotes equality up to a normalizing scale factor. Using the equations relating

[1]Notation: The preceding superscript for vectors (e.g., G in ${}^G\mathbf{p}$) 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 $\{X\}$. ${}^X\mathbf{p}_Y$ 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.

the points' coordinates in $\{C\}$ with the points' coordinates in $\{A\}$, ${}^{C}\mathbf{p}_i = {}^{C}_{A}\mathbf{R}\,{}^{A}\mathbf{p}_i + {}^{C}\mathbf{p}_A$, $i = 1, 2$, we obtain:

$$ {}^{C}\mathbf{n}_{\pi_C} \sim {}^{C}_{A}\mathbf{R}\left({}^{A}\mathbf{p}_1 \times {}^{A}\mathbf{p}_2\right) + {}^{C}\mathbf{p}_A \times \left({}^{C}_{A}\mathbf{R}\left({}^{A}\mathbf{p}_2 - {}^{A}\mathbf{p}_1\right)\right) \tag{2.1} $$

We now note that the term ${}^{A}\mathbf{p}_1 \times {}^{A}\mathbf{p}_2$ can be written as:

$$ {}^{A}\mathbf{p}_1 \times {}^{A}\mathbf{p}_2 = ||{}^{A}\mathbf{p}_1||\,||{}^{A}\mathbf{p}_2||\sin\langle {}^{A}\mathbf{p}_1, {}^{A}\mathbf{p}_2\rangle {}^{A}\mathbf{n}_{\pi_A} $$

where $\mathbf{n}_{\pi_A}$ is the unit vector normal to the plane defined by the line and the origin of $\{A\}$ (see Fig. 2.1). Using this result, and expressing the quantities in the above equation with respect to the global reference frame, $\{G\}$, we obtain:

$$ \begin{aligned} {}^{C}\mathbf{n}_{\pi_C} \sim\ & {}^{C}_{G}\mathbf{R}\Big(||{}^{A}\mathbf{p}_1||\,||{}^{A}\mathbf{p}_2||\sin\langle {}^{A}\mathbf{p}_1, {}^{A}\mathbf{p}_2\rangle {}^{G}\mathbf{n}_{\pi_A} + \\ & \left({}^{G}\mathbf{p}_C - {}^{G}\mathbf{p}_A\right) \times \left({}^{G}_{A}\mathbf{R}\,{}^{A}\mathbf{p}_1 - {}^{G}_{A}\mathbf{R}\,{}^{A}\mathbf{p}_2\right)\Big) \\ \sim\ & {}^{C}_{G}\mathbf{R}\left({}^{G}\mathbf{n}_{\pi_A} + \left({}^{G}\mathbf{p}_C - {}^{G}\mathbf{p}_A\right) \times {}^{G}\mathbf{v}_\ell\right) \end{aligned} \tag{2.2} $$

where

$$ {}^{G}\mathbf{v}_\ell = \frac{1}{||{}^{A}\mathbf{p}_1||\,||{}^{A}\mathbf{p}_2||\sin\langle {}^{A}\mathbf{p}_1, {}^{A}\mathbf{p}_2\rangle}\left({}^{G}_{A}\mathbf{R}\,{}^{A}\mathbf{p}_1 - {}^{G}_{A}\mathbf{R}\,{}^{A}\mathbf{p}_2\right) $$

Let us now examine the terms appearing in (2.2). First, we have the camera position and rotation with respect to the global frame, ${}^{G}\mathbf{p}_C$ and ${}^{C}_{G}\mathbf{R}$. Second, we have the position of the origin of the anchor frame, ${}^{G}\mathbf{p}_A$, which is a known constant. Third, we have the vector ${}^{G}\mathbf{n}_{\pi_A}$, which is the unit vector formed by the origin of $\{A\}$ and the line, expressed with respect to the global frame. This vector depends on the line's position and orientation in space, and therefore it will constitute part of the line parameters. Finally, turning our

attention to the vector ${}^G\mathbf{v}_\ell$, we note that this vector is a linear combination of ${}^A\mathbf{p}_1$ and ${}^A\mathbf{p}_2$ (expressed in $\{G\}$). Since both vectors lie in the plane normal to ${}^G\mathbf{n}_{\pi_A}$, ${}^G\mathbf{v}_\ell$ must also lie in this plane, and thus can be written as a linear combination of two basis vectors within this plane. We can therefore write ${}^G\mathbf{v}_\ell$ as:

$$ {}^G\mathbf{v}_\ell = a_1 {}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_1 + a_2 {}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_2 \tag{2.3} $$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are two known constant vectors. Note that, for any choice of these two vectors, both ${}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_1$ and ${}^G\mathbf{n}_{\pi_A} \times \mathbf{w}_2$ lie in the plane normal to $\mathbf{n}_{\pi_A}$, and therefore these are two valid basis vectors, assuming they are linearly independent. In our work, to ensure linear independence, we select $\mathbf{w}_i = {}^G_A\mathbf{R}\left({}^A\mathbf{p}_i \times {}^A\mathbf{n}_{\pi_A}\right)$, $i = 1, 2$.

To summarize, a 3D line in our work is parameterized by the unit vector ${}^G\mathbf{n}_{\pi_A}$ and the parameters a_1, a_2. Since the unit vector has two degrees of freedom, this parameterization corresponds to a total of four degrees of freedom, as required for a 3D line. We note that since the norms of the vectors ${}^A\mathbf{p}_1$ and ${}^A\mathbf{p}_2$ (i.e., the "depth" of these points with respect to the anchor frame $\{A\}$) appear in the denominator of the expression for ${}^G\mathbf{v}_\ell$, the parameters a_1 and a_2 have units of "inverse depth", hence the name of our proposed parameterization.

Given this line parameterization, the vector normal to the plane defined by the camera optical center and the line is given by equations (2.2) and (2.3). How this normal vector is used in order to construct a measurement model in our EKF estimator is explained in the next section. As a final remark, we note that to ensure a minimal representation for the errors, the error-vector for a given line parameterization $\mathbf{f}_L = [{}^G\mathbf{n}_{\pi_A}^T \;\; a_1 \;\; a_2]^T$ is defined

as:

$$\tilde{\mathbf{f}}_L = \begin{bmatrix} k_1 & k_2 & \tilde{a}_1 & \tilde{a}_2 \end{bmatrix}^T$$

where $\tilde{a}_1$ and $\tilde{a}_2$ represent the errors in the estimates of a_1 and a_2, while k_1 and k_2 are used to define a minimal representation for the error in the unit vector ${}^G\mathbf{n}_{\pi_A}$. Specifically, to a first-order approximation, the relationship between the true and estimated unit vectors is given by:

$$ {}^G\mathbf{n}_{\pi_A} = {}^G\hat{\mathbf{n}}_{\pi_A} + \mathbf{B}_{\mathbf{n}_{\pi_A}} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{2.4}$$

where $\mathbf{B}_{\mathbf{n}_{\pi_A}}$ is a matrix whose columns are chosen to be two orthogonal vectors perpendicular to ${}^G\hat{\mathbf{n}}_{\pi_A}$.

2.4 Estimator formulation

We now turn to the problem of using line measurements to track the motion of a platform equipped with an IMU and a RS camera in an unknown environment. The EKF-based estimator that we employ to track this state is a modification of the "hybrid" filter proposed in [45].

In prior work, the hybrid estimator was employed in conjunction with point-feature measurements only. Here, in addition to point features, we utilize the observations of straight-line features in the environment. Since the details of the hybrid estimator have been presented in prior work, we here briefly describe the structure of the estimator (see Algorithm 1), and refer the reader to [45] for further details.

Our goal is to estimate the pose of the moving platform with respect to a global coordinate frame $\{G\}$. To this end, we affix a "body" coordinate frame, $\{B\}$, to the IMU, and track the motion of this frame with respect to $\{G\}$. The origin of the body frame is chosen as the point where the three accelerometer axes intersect while its orientation is determined based on the camera frame $\{C\}$. In our formulation the rotation of $\{B\}$ with respect to $\{C\}$ is defined to be a known, constant matrix ${}^{C}_{B}\mathbf{R}$. In our implementation the value of this constant matrix is selected to be prior estimate of the camera-to-IMU rotation. As a result the axes of $\{B\}$ are "close" to axes of the IMU sensors, which we seek to calibrate using self-calibration.

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

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{\mathbf{c}}^T & \mathbf{x}_{B_1}^T & \cdots & \mathbf{x}_{B_m}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_s^T \end{bmatrix}^T \tag{2.5}$$

where $\mathbf{x}_{E_k}$ is the "evolving state" of the IMU, comprising the current body-frame position, velocity, orientation, as well as the time-varying IMU biases; $\mathbf{x}_{\mathbf{c}}$ is the vector of parameters we seek to calibrate, which is defined in 2.14; the states $\mathbf{x}_{B_j}$, $j = 1 \ldots m$ 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. These features include *both* points, which are represented in inverse-depth parameterization (IDP) [53], and straight lines, which are represented in the TPIDP parameterization described in the preceding section.

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 and line features, which are used 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. For this purpose, the multi-state-constraint method of [44, 54] 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 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 [44].

2.5 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 addi-

tion to random noise). For a full description of the sources of errors and their modeling, the reader is referred to [71]. 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:

$$a_{m_i} = s_i {}^B\mathbf{u}_i^T {}^B\mathbf{a}_s + b_{a_i} + n_{a_i} \qquad i = 1, 2, 3 \tag{2.6}$$

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 [71].

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 (2.6)). 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[2]. Therefore, the evolving state of the EKF, $\mathbf{x}_E$, is given by:

$$\mathbf{x}_E = \begin{bmatrix} {}^B_G\bar{\mathbf{q}}^T & {}^G\mathbf{p}_B^T & {}^G\mathbf{v}_B^T & \mathbf{b}_g^T & \mathbf{b}_a^T \end{bmatrix} \tag{2.7}$$

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 (2.5)). 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 (2.14)).

[2]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.

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:

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

These are subsequently be used to propagate the estimate of the evolving state via numerical integration, as described in [44]. Moreover, the covariance matrix of the EKF is propagated. For this step, we first compute the Jacobian matrices $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Gamma}_k$, which describe the relationship between the evolving-state errors in propagation:

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

where $\mathbf{w}_k$ is the process-noise error, modeled as zero-mean Gaussian with covariance matrix $\mathbf{Q}_k$. The matrices $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Gamma}_k$ are computed analytically, similarly to [44]. 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:

$$
\mathbf{P}_{k+1} = \begin{bmatrix} \boldsymbol{\Phi}_k & \boldsymbol{\Gamma}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{P}_k \begin{bmatrix} \boldsymbol{\Phi}_k & \boldsymbol{\Gamma}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \mathbf{Diag}(\mathbf{Q}_k, \mathbf{0}) \tag{2.8}
$$

2.6 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 $\mathbf{x_c}_{IMU}$ form the parameter vector $\mathbf{x_c}$ in (2.5). 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 [43]. 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

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

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:

$$\mathbf{z} = \mathbf{h}({}^C\mathbf{p}_f(t_n)) + \mathbf{n} \tag{2.9}$$

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 [28]. Denoting ${}^C\mathbf{p}_f(t_n) = [{}^Cx_f \;\; {}^Cy_f \;\; {}^Cz_f]^T$, the image projection is given by:

$$\mathbf{h}\left({}^C\mathbf{p}_f\right) = \mathbf{p}_c + \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix} \begin{bmatrix} u_d \\ v_d \end{bmatrix} \tag{2.10}$$

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

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = d \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2t_1 uv + t_2(u^2 + v^2 + 2uv) \\ t_1(u^2 + v^2 + 2uv) + 2t_2 uv \end{bmatrix} \tag{2.11}$$

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

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^C z_f} \begin{bmatrix} {}^C x_f \\ {}^C y_f \end{bmatrix} \tag{2.12}$$

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:

$${}^C\mathbf{p}_f(t_n) = {}^C_B\mathbf{R}\, {}^B_G\mathbf{R}(t_n) \left({}^G\mathbf{p}_f - {}^G\mathbf{p}_B(t_n) \right) + {}^C\mathbf{p}_B \tag{2.13}$$

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:

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{x}_{\mathbf{c}_{IMU}}^T & {}^C\mathbf{p}_B^T & \mathbf{p}_c^T & a_u & a_v & k_1 & k_2 & k_3 & t_1 & t_2 & t_r & t_d \end{bmatrix}^T \tag{2.14}$$

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.

2.6.1 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 (2.13), 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 [26, 42], or higher-order parametric forms [27, 58]). 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 (2.9) is defined as:

$$\mathbf{r} = \mathbf{z} - \mathbf{h}\left({}^{C}_{B}\mathbf{R}\,{}^{B}_{G}\hat{\mathbf{R}}(\hat{t}_n)\left({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_B(\hat{t}_n)\right) + {}^{C}\hat{\mathbf{p}}_B\right) \tag{2.15}$$

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 (2.15), to obtain:

$$\mathbf{r} \simeq \mathbf{H}_{\boldsymbol{\theta}} \tilde{\boldsymbol{\theta}}_B(\hat{t}_n) + \mathbf{H}_{\mathbf{p}} {}^G\tilde{\mathbf{p}}_B(\hat{t}_n) + \mathbf{H}_{\mathbf{c}} \tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}} \tilde{\mathbf{f}} + \mathbf{n} \tag{2.16}$$

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 $\mathbf{H}_{\mathbf{c}}$ is the Jacobian with respect to $\mathbf{x}_{\mathbf{c}}$. For details on the definition of the orientation error, $\tilde{\boldsymbol{\theta}}_B$, and the computation of the time-related Jacobians, please refer to [43].

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:

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

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.

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 unmodeled reprojection errors are small.

Using these approximations in (2.16), we obtain the following linearized equation for the measurement residual:

$$\begin{aligned}\mathbf{r} \simeq\ & \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_B(t+\hat{t}_d) + \mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{p}}_B(t+\hat{t}_d) + \frac{n\hat{t}_r}{N}\mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{v}}_B(t+\hat{t}_d) \\ & + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}}\tilde{\mathbf{f}} + \mathbf{n} \end{aligned} \tag{2.17}$$

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.

2.6.2 Line measurement model

We now describe the formulation of the measurement model that we employ for the straight-line features detected in the images. In prior work, the standard practice is to employ line fitting in the images to obtain the equations of the lines in the images, and to subsequently relate these equations to the configuration of the line and camera in space.

However, when a moving RS camera observes straight lines, these are no longer guaranteed to project into straight lines in the images. Therefore, our approach completely forgoes the line-fitting step. Instead, all the pixels that are deemed to belong to a straight line (see Section 3.5.2 for a description of our line-detection strategy) are directly used to define measurement residuals for the EKF update.

Specifically, let us consider an image point that belongs to the projection of a 3D line, and lies in the i-th row of the RS image. If we denote the normalized coordinates of the point as (u_i, v_i), then the following equation is satisfied:

$$\begin{bmatrix} u_i & v_i & 1 \end{bmatrix} {}^{C_i}\mathbf{n}_{\pi_C} = 0$$

where C_i is the camera pose at time t_i, i.e., at the time instant the i-th image row was recorded, and ${}^{C_i}\mathbf{n}_{\pi_C}$ is defined by equations (2.2) and (2.3). We note that the above equation involves the noise-less image projection coordinates, as well as the true line parameters. In any real-world estimation problem, however, the measurements will be corrupted by noise, and only estimates of the line parameters are available. Therefore, using these quantities, we obtain a *residual*:

$$\begin{aligned} r_i &= \begin{bmatrix} \mathbf{z}_i^T & 1 \end{bmatrix} {}^{C_i}\hat{\mathbf{n}}_{\pi_C} \\ &= \begin{bmatrix} \mathbf{z}_i^T & 1 \end{bmatrix} {}^{C}_{G}\hat{\mathbf{R}}(\hat{t}_i) \left({}^{G}\hat{\mathbf{n}}_{\pi_A} + \left({}^{G}\hat{\mathbf{p}}_C(\hat{t}_i) - {}^{G}\mathbf{p}_A \right) \times {}^{G}\hat{\mathbf{v}}_\ell \right) \end{aligned} \tag{2.18}$$

where

$${}^{G}\hat{\mathbf{v}}_\ell = \hat{a}_1 {}^{G}\hat{\mathbf{n}}_{\pi_A} \times \mathbf{w}_1 + \hat{a}_2 {}^{G}\hat{\mathbf{n}}_{\pi_A} \times \mathbf{w}_2$$

In the above, $\mathbf{z}_i$ represents the normalized-image measurement vector,

$$\mathbf{z}_i = \begin{bmatrix} u_i \\ \\ v_i \end{bmatrix} + \boldsymbol{\eta}_i$$

where $\boldsymbol{\eta}_i$ is the image measurement noise, modeled as zero-mean, Gaussian, with covariance matrix $\sigma^2\mathbf{I}_2$, and $\hat{\cdot}$ denotes the estimated value of a quantity. By employing linearization, we can express the residual, up to a first order approximation, as a function of the errors in the state estimates, the line-parameter estimates, and the measurement noise:

$$r_i \simeq \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_B(\hat{t}_i) + \mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{p}}_B(\hat{t}_i) + \mathbf{H}_{\mathbf{f}}\tilde{\mathbf{f}}_L + \mathbf{H}_{\mathbf{c}}\tilde{\mathbf{x}}_{\mathbf{c}} + \boldsymbol{\Gamma}_i\boldsymbol{\eta}_i \tag{2.19}$$

where $\tilde{\boldsymbol{\theta}}_B(\hat{t}_i)$ and ${}^G\tilde{\mathbf{p}}_B(\hat{t}_i)$ are the errors in the orientation and position estimates at time instant $\hat{t}_i$, $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{p}}$ are the corresponding Jacobians, $\tilde{\mathbf{f}}_L$ is the error in the estimates of the line parameters and $\mathbf{H}_{\mathbf{f}}$ is the corresponding Jacobian, $\tilde{\mathbf{x}}_{\mathbf{c}}$ is the error in the estimates of the calibration parameters and $\mathbf{H}_{\mathbf{c}}$ is the corresponding Jacobian and $\boldsymbol{\Gamma}_i$ is the Jacobian of r_i with respect to $\boldsymbol{\eta}_i$. Having defined a residual in (2.18) and its linearized expression in (2.19), we can now directly apply the method of [45] for performing an EKF update.

2.7 Maximum-a-posteriori self-calibration

We now describe a maximum-a-posteriori (MAP) estimator used to process both IMU and image measurements for self-calibration when large initial errors do not allow successful use of the EKF method described in the preceding section. In this formulation the history of IMU states, estimates of feature parameters (IDP for points and TPIDP for lines) and sensor model parameters are jointly optimized. Specifically, we assume that there are N images and inertial measurements during the history of recorded images are

available. The MAP estimates for all these quantities are determined by minimizing the following cost function:

$$
\begin{aligned}
\mathbf{J} = & \ \mathbf{J}_1 + \mathbf{J}_2 + \mathbf{J}_3 + \mathbf{J}_4 \\
= & \ \|\mathbf{x}_{B_1} - \mathbf{x}_{B_p}\|_{\mathbf{Q}_p} + \ \|\mathbf{x}_c - \mathbf{x}_{c_o}\|_{\mathbf{Q}_c} + \sum_{\ell,\ j} \|\mathbf{z}_\ell^{(j)} - \mathbf{h}\left(\mathbf{f}_j,\ \mathbf{x}_{B_\ell},\ \mathbf{x}_c\right)\|_{\mathbf{R}_\ell^{(j)}} \\
& + \sum_{\ell=1}^{\ell=N} \|\mathbf{x}_{B_{\ell+1}}\left(t_{\ell+1} + t_d\right) - \mathbf{\Phi}\left(\mathbf{x}_{B_\ell}\left(t_\ell + t_d\right), {}^B\boldsymbol{\omega}_m, {}^B\mathbf{a}_m\right)\|_{\mathbf{Q}_{d_\ell}}
\end{aligned} \tag{2.20}
$$

In the equation above, notation $\|\mathbf{v}\|_{\mathbf{Q}}$ denotes the Mahalanobis norm $\mathbf{v}^T\mathbf{Q}^{-1}\mathbf{v}$. There are four terms involved in the overall cost function and each part corresponds to one type of information available to the estimation problem at hand.

- Terms $\mathbf{J}_1$ and $\mathbf{J}_2$ convey the prior information that is available to the system. Specifically, the term $\mathbf{J}_1$ is the prior on the initial state of the IMU, with the prior estimate and its covariance denoted by $\mathbf{x}_{B_p}$ and $\mathbf{Q}_p$, respectively. Typically estimates of the initial pose and velocity of the system at the start of a trajectory may be available if it is known that the system is kept static initially. On the other hand, $\mathbf{J}_2$ expresses the prior information for the calibration parameters. The prior estimates of these quantities may be obtained from earlier calibrations, CAD models, and/or use of "ideal" sensor models (e.g., assuming perfect orthogonality of IMU axes, zero biases, and unit scale factors).

- Term $\mathbf{J}_3$ expresses the weighted measurement residual for both line and point features. We use the index ℓ to refer the body pose that observes the feature and j.

- The last term, $\mathbf{J}_4$, expresses the constraints due the process model of IMU. This

includes the weighted difference between the state estimates and predicted estimates at $\mathbf{t}_{\ell+1}$ by integrating IMU measurements. Due to temporal misalignment between IMU and image timestamp, the time offset t_d is involved in term $\mathbf{J}_4$. The Jacobians with respect to the IMU states and the time offset t_d are computed similarly to [43,44]

In order to minimize the cost function above, we employ Gauss-Newton iterative minimization. The information matrix $\mathbf{\Lambda}$ resulting from all constraints is built at each iteration, and the Gauss-Newton system is solved via Cholesky factorization.

Due to the higher computational cost of the iterative minimization algorithm compared to the EKF approach, we only employ this this batch algorithm for estimator initialization. Once the minimization has converged, the estimates of the IMU states and sensor parameters are used to initialize the hybrid filter. The EKF covariance matrix is also computed given the available Cholesky factorization. Specifically, let us assume that the state vector defined in the Gauss-Newton minimization is partitioned as $\mathbf{x}_{MAP} = [\mathbf{x}_d^T \;\; \mathbf{x}_k^T]^T$, and we seem to compute the covariance of $\mathbf{x}_k$. Then, if the upper triangular Cholesky factor corresponding to $\mathbf{x}_{BA}$ is given by:

$$\mathbf{R}_{MAP} = \begin{bmatrix} \mathbf{R}_{dd} & \mathbf{R}_{dk} \\ \mathbf{0} & \mathbf{R}_{kk} \end{bmatrix} \tag{2.21}$$

The covariance matrix of $\mathbf{x}_k$ is obtained as $\mathbf{P}_{kk} = \mathbf{R}_{kk}^{-1}\mathbf{R}_{kk}^{-T}$. Once the hybrid filter is appropriately initialized, it begins to process inertial and visual data as described in Algorithm 1.

2.8 Simulation Results

In this section we present simulation results which demonstrate the properties of the proposed line parameterization compared to alternatives by using the metric proposed in in [66]. Based on this result, we further evaluate the performance of the proposed visual-inertial odometry estimator which processes both point and line measurements by adopting different parameterization for line.

2.8.1 Comparison of line parameterizations

We first examine the linearity characteristics of the proposed line parameterization. We note that, as discussed in [53, 66], one of the key properties a state parameterization must have, when used in an EKF estimator, is that it has to result in measurement models with "small" nonlinearity. This is necessary, as large nonlinearities result in non-Gaussian errors, and violate the small linearization-error assumptions of the EKF, leading to poor performance. Hence, we here compare the linearity of the proposed TPIDP against the following parameterizations: (i) the orientation-depth parameterization (ODP) employed in [39], (ii) the orientation-depth parameterization with the depth represented by its inverse, which we term orientation-inverse depth (OIDP), (iii) the Cayley parameterization of [80], (iv) the Roberts parameterization [61], and (v) the orthonormal representation (Ortho) of [6]. Note that while additional parameterizations have been proposed in the literature (e.g., Plücker coordinates, two-point parameterizations), we are only interested in minimal-error-representation parameterizations, due to their numerical advantages in EKF-based estimation.

The metric that we use to measure linearity is the measurement-linearity index defined in [66]. Specifically, if we denote by $\mathbf{y}$ the $n \times 1$ vector containing the line parameters and the poses from which this line was observed, the linearity index is defined as:

$$\lambda = \|\epsilon\|_2, \ \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_q \end{bmatrix}, \ \epsilon_i \simeq \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} H_{ijk} P_{jk} \tag{2.22}$$

where q is the number of pixel observations corresponding to the line, $H_{ijk} = \frac{\partial^2 r_i}{\partial \mathbf{y}_j \partial \mathbf{y}_k}$ is the second-order derivative of the measurement residual function (3.9), and P_{jk} is the (j,k) element of the joint covariance matrix of $\mathbf{y}$. Intuitively, the above metric evaluates the magnitude of the second-order and higher terms in the measurement model, which are ignored during EKF's linearization.

To evaluate the linearity of the different parameterizations, we perform Monte-Carlo simulations. In each trial, we randomly generate a line feature and m camera poses, with m randomly selected between 8 and 15. The joint covariance matrix of the camera poses is selected to be identical to the joint covariance matrix observed at a certain time in one of our real-world experiments. For each randomly generated configuration we generate simulated line-pixel measurements, and employ them to estimate the line parameters using least-squares minimization. Subsequently, following the approach in [48], we compute the joint covariance matrix of the line and camera poses, and use it to compute the linearity index λ as shown in (2.22). In Table 2.1 we present the average linearity index, averaged over the 6000 Monte-Carlo trials, for all the compared parameterizations. Recall that in

Table 2.1: Linearity Index of Different Line Parameterizations

Parameterization	Average Linearity Index
TPIDP	0.000049
OIDP	0.0013
ODP	0.0149
Roberts	0.0045
Ortho	0.0017
Cayley	0.0055

this table smaller values indicate smaller nonlinearity, and are preferable. We can clearly see that the linearity index associated with the TPIDP parameterization is substantially smaller than the index associated with all other parameterizations. Specifically, it is approximately 3.7 percent of the linearity index of the second-best parameterization, which is the OIDP. This result demonstrates that the TPIDP results in a measurement model that is "closer to linearity", and thus we expect to obtain improved performance when using this parameterization in an EKF-based estimator. Moreover, this result indicates that the covariance matrix computed by linearization of the measurement models using TPIDP provides a more accurate description of the uncertainty.

2.8.2 Comparison of the hybrid EKF with different parameterizations

We next examine the effect of using straight-line measurements, and parameterizing these lines in different ways, in the hybrid EKF estimator. To demonstrate the effect

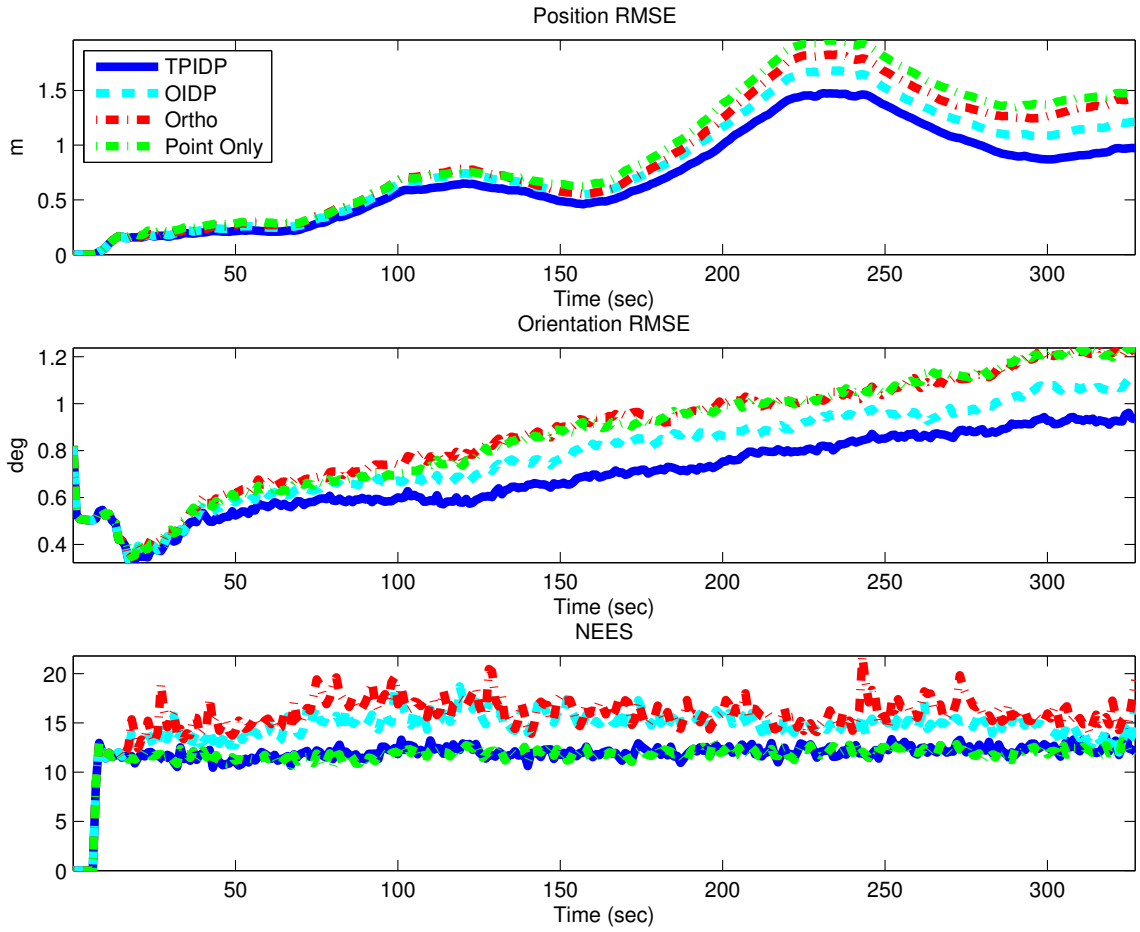

Figure 2.2: Hybrid EKF simulations: average NEES and RMSE over 100 Monte-Carlo trials

caused by different choice of line parameterization, we assume that all calibration parameters in sensors' model except the time-varying IMU biases are known in advance. To this end, we perform Monte-Carlo simulations in a simulation environment that emulates a real-world dataset. Specifically, in each Monte-Carlo trial, we generate a ground-truth trajectory that follows the trajectory of the actual dataset, and in each image we generate simulated point and straight-line feature measurements with characteristics (noise levels, feature numbers, track lengths) as in the real-world dataset. The trajectory's length was

402 m, lasting about 328 s. The average number of point features per image is 30, while the average number of line features is 15, and each line consists of 20 pixels on average. We process the data by the following methods: (i) the hybrid EKF estimator that uses point features only, (ii) the proposed hybrid EKF that uses both points and lines, using the TPIDP representation for the lines included in the state, (iii) the hybrid EKF using points and lines with the OIDP representation, and (iv) the hybrid EKF using points and lines with the Orthogonal representation of [6]. The three line parameterizations used here are the best three parameterizations in terms of linearity, as shown in the preceding tests.

The metrics we use to evaluate the performance of the different methods are (i) the root-mean square error (RMSE) of the orientation and position estimates, and (ii) the normalized estimation error squared (NEES) for the IMU state consisting of the orientation, position, and velocity. The RMSE gives us a measure of the accuracy of the estimator, while the NEES provides us with a measure of consistency [4, 44]. Specifically, for zero-mean Gaussian errors the NEES should equal the dimension of the error-state, i.e., 9 in this case. Larger values indicate that there exist unmodeled errors (e.g., linearization errors) in the estimator, and that the covariance matrix reported by the EKF underestimates the actual magnitude of the estimation errors. Examining both metrics provides us with a more complete picture of the estimator's performance.

Both the RMSE and NEES are averaged over 100 Monte-Carlo trials, and the results are plotted over time in Fig. 2.2. Table 2.2 lists the average values over all Monte-Carlo trials and all time instants. From these results, we can first observe that, as expected, using line features in addition to point features provides additional information to the

Table 2.2: Average RMSE and NEES for Fig. 2.2

Simulation Settings	Position RMSE (m)	Orientation RMSE (deg)	NEES
TPIDP	0.7059	0.6964	11.8096
OIDP	0.8192	0.7875	14.4869
Ortho	0.8947	0.8793	15.5772
Points Only	0.9554	0.8699	11.6510

estimator, and leads to lower estimation errors. Moreover, we can clearly see that the TPIDP parameterization results in both lower estimation errors, as well as in lower NEES, compared to the alternative parameterizations examined. These results agree with the linearity-index results presented earlier, and demonstrate the advantages of the proposed TPIDP representation for EKF-based estimation using line features.

We note that the NEES values computed using all three line parameterizations, as well as for the point-only EKF, are higher than the "ideal" value of 9. While TPIDP performs better than the other parameterizations, it also appears to be mildly inconsistent. We attribute this result primarily to the small number of features used. Being able to track only a small number of features, which is typical of low-texture indoor environments (such as office environments with textureless walls), results in increased errors and thus more pronounced nonlinearities in all the methods tested.

Table 2.3: Simulations: RMS errors of calibration parameters $\mathbf{x}_{IMU}$

Time (sec)	Simulation Settings	$\mathbf{b}_g$ (rad/sec)	$\mathbf{b}_a$ (m/sec^2)	$\mathbf{T}_g$	$\mathbf{T}_a$	$\mathbf{T}_s$ (rad/sec/g)
0	All approaches	0.0162	0.3126	0.0603	0.0605	0.0012
8.4	Method 1	0.0052	0.1432	0.0080	0.0195	0.00086
	Method 2	0.0048	0.1185	0.0054	0.0166	0.00074
	Method 3	0.0063	0.2393	0.0095	0.0311	0.0010
	Proposed method	0.0044	0.1071	0.0051	0.0145	0.00068
40	Method 1	0.0026	0.0363	0.0028	0.0053	0.00036
	Method 2	0.0024	0.0356	0.0022	0.0050	0.00033
	Method 3	0.0025	0.0508	0.0028	0.0071	0.00035
	Proposed method	0.0020	0.0327	0.0018	0.0046	0.00029

2.8.3 Self-calibration using line feature

Next, we evaluate the performance of self-calibration when using straight-line measurements. In this simulation data generation is performed the same way as in the previous simulation, but for a 40-second-long trajectory hat involves significant rotation and translation. For the parameter values in the sensor models of the camera and IMU, in each trial we draw errors from zero-mean Gaussian distributions and add them to known nominal values. To demonstrate the effectiveness of the proposed algorithm even when only coarse estimates of sensor parameters are available, the magnitude of errors that are added to the nominal values of the parameters are larger than what we typically find in practice. The characteristics of the generated features (both points and lines) are also the same as in the

Table 2.4: Simulations: RMS errors of calibration parameters ${}^{C}\mathbf{p}_B^T$ and t_d

Time (sec)	Simulation Settings	${}^{C}\mathbf{p}_B$ (cm)	t_d (msec)
0	All approaches	0.0605	0.0012
8.4	Method 1	0.0195	0.00086
	Method 2	0.0166	0.00074
	Method 3	0.0311	0.0010
	Proposed method	0.0145	0.00068
40	Method 1	0.0053	0.00036
	Method 2	0.0050	0.00033
	Method 3	0.0071	0.00035
	Proposed method	0.0046	0.00029

previous simulation, with an average number of 100 points and 20 lines per image.

The datasets are processed by the following four approaches: (i) the hybrid filter using point features only (referred to as 'Method 1'), (ii) the hybrid filter using point features only with batch MAP initialization (referred to as 'Method 2'), (iii) the hybrid filter using both point and line features (referred to as 'Method 2'), and (iv) the hybrid filter using both point and line features with batch MAP initialization (referred to as 'Proposed Method'). To demonstrate the performance of calibration accuracy using these 4 different approaches, we compare the RMS errors of all calibration parameters over 50 time Monte-Carlo trials as shown in Table 2.3, Table 2.4, Table 2.5 and Table 2.6. In all tables, 0 sec corresponds to the initial errors, 8.4 sec corresponds to the errors at the end of

Table 2.5: Simulations: RMS errors of principal points, focal length and image readout time of the calibration parameters $\mathbf{x}_{cam}$

Time (sec)	Simulation Settings	a_u (pix.)	a_v (pix.)	$\mathbf{u}_o$ (pix.)	$\mathbf{v}_o$ (pix.)	t_r (msec)
0	All approaches	5.1123	4.4315	5.3862	5.1604	4.4
8.4	Method 1	1.2655	1.8116	2.0804	2.0669	0.31
	Method 2	0.7362	0.9990	1.4920	1.4653	0.17
	Method 3	1.3529	2.0570	2.6034	2.6805	0.43
	Proposed method	0.5636	0.7973	1.4889	1.4640	0.12
40	Method 1	0.5440	0.8337	0.5499	0.5885	0.11
	Method 2	0.4416	0.5137	0.5287	0.5441	0.09
	Method 3	0.4617	0.8679	0.5603	0.6065	0.09
	Proposed method	0.3399	0.4032	0.4609	0.4848	0.07

batch minimization (when used), and 40 sec corresponds to the final errors. As we can see from the final errors of all calibration parameters, approaches that rely on bundle-adjustment-based initialization achieve better accuracy for all calibration parameters. We also notice that due to the increased linearization errors of line measurement model, the performance of self-calibration approach directly using the EKF-based estimator (without iterative minimization for initialization) is lower. This clearly shows the advantage of a bundle-adjustment-based approach in handling the greater nonlinearities encountered when line measurements and large initial calibration errors exist. Moreover, the improvement in calibration accuracy ultimately leads to improvement of both the accuracy and consistency of the hybrid filter as shown in Table 2.7.

Table 2.6: Simulations: RMS errors of radial distortion and tangential distortion coefficients of the calibration parameters $\mathbf{x}_{cam}$

Time (sec)	Simulation Settings	k_1	k_2	k_3	t_1	t_2
0	All approaches	0.0975	0.0890	0.0969	0.00091	0.00087
8.4	Method 1	0.0097	0.0391	0.0500	0.00076	0.00061
	Method 2	0.0067	0.0277	0.0364	0.00064	0.00045
	Method 3	0.0082	0.0361	0.0487	0.00073	0.00064
	Proposed method	0.0060	0.0242	0.0326	0.00047	0.00036
40	Method 1	0.0048	0.0217	0.0295	0.00039	0.00030
	Method 2	0.0042	0.0194	0.0263	0.00028	0.00027
	Method 3	0.0026	0.0116	0.0149	0.00034	0.00029
	Proposed method	0.0026	0.0115	0.0150	0.00025	0.00023

It is also interesting to examine the result of one typical trial as shown in Fig. 2.3. We here compare the result from the hybrid filter using point feature only, with the only difference being the way the estimator is initialized. We notice that the errors of the calibration parameters quickly decrease in the first few seconds, and the errors are all within the 3-sigma bound reported by either EKF-based approach or the bundle-adjustment approach. These results also corroborate the fact that all calibration parameters are identifiable for sufficiently exciting motion.

Table 2.7: Average RMSE and NEES of pose in Calibration dataset

Simulation Settings	Position RMSE (m)	Orientation RMSE (deg)	NEES
Method 1	0.0571	0.5008	7.7590
Method 2	0.0517	0.4217	6.4032
Method 3	0.0926	0.7222	32.8058
Proposed method	0.0440	0.3807	6.6688

2.9 Real-World Experiment

2.9.1 Use of line features in point-feature poor environments

In addition to the simulation tests, we also tested our proposed approach in a real-world experiment, which was conducted using the sensors of a Nexus 4 device. The experiment was conducted in an indoor office area of the UCR Engineering Building, where line features are the most visually dominant ones. The device was hand-held by a person walking the halls of three floors of the building during the experiment. For point-feature extraction and matching, Shi-Tomasi features are used [64], and matched by normalized cross-correlation. To detect and match line features, we first use the Canny edge detector [8] to identify edge pixels. The normalized coordinates of these pixels are computed using the pre-computed calibration parameters, and subsequently we perform partial RS compensation. Specifically, we employ the rotation estimates from the IMU to remove the rotational component of the RS distortion. The resulting compensated coordinates are employed for straight-line detection using a split-and-merge approach. We point out that

compensation is performed *only* to aid in detecting straight lines in the environment, and the compensated coordinates are *not* used for EKF updates. For line matching, a template is generated at the midpoint of each line, and matching is performed by normalized-cross correlation.

In this experiment, an average of 29.71 point features and 16 line segments are extracted per image. The trajectory length of the experiment is approximately 400 m. The IMU sample rate is 200 Hz, while the images are captured at 22 Hz (sample images are shown in Fig. 2.5). All the data are post-processed off-line on a desktop computer, to enable comparing the performance of different approaches. In this test, we are comparing (i) the hybrid EKF estimator using point features only (referred to as Points only), (ii) the proposed hybrid EKF using point and line features concurrently (referred to as Proposed method), and (iii) the hybrid EKF using point and line features where *only* lines are processed as if they are acquired from a GS camera (referred to as Lines without RS model). Since the ground truth for the entire trajectory is not available, the fact that the trajectory starts and ends at the same point is used to evaluate the performance of the algorithms.

The 3D trajectory estimated by each approach is shown in Fig. 2.6. Moreover, the computed final position errors for each of the methods are 1.48 m for the proposed method using both points and lines (corresponding to 0.37% of the traveled distance), 3.05 m for the hybrid EKF that uses point measurements only and 4.05 m for the hybrid EKF that uses both points and lines with partial RS compensation. Thus we can clearly see that processing line features, especially in environments that are poor in point features and rich in straight lines, can lead to measurable performance gains in vision-aided inertial

navigation, compared to algorithms that only employ point features. Furthermore, we notice that simply performing RS compensation, and treating the resulting measurements as if they come from a global shutter camera, leads to worse performance than the proposed approach. This is due to the fact that RS compensation can only be reliably performed for the rotational effects, while the effects of the camera translational motion during the image readout time cannot be exactly compensated for.

2.9.2 Self-calibration using line features

In this section we demonstrate that improved performance can be achieved by using line measurements in calibration. In this experiment a Nexus 4 phone is used to collect the data. The experiment was conducted in an indoor area of the UCR Science Library, where the point features are abundant and line features are also prevalent. This experiment lasted about 11 min, and the trajectory length is approximately 900 m. All the data are post-processed off-line on a desktop computer, to enable comparing the performance of different approaches. The prior estimates of calibration parameters are initialized as follows: the camera intrinsics are set to values from previous calibration; the image readout time is set equal to the image period; the prior of time-offset between IMU and camera was set to zero; the camera-to-IMU translation is set to zero; both the g-sensitivity matrix $\mathbf{T}_s$ and IMU biases are initialized to zero, and finally the prior estimates for the misalignment matrices $\mathbf{T}_g$ and $\mathbf{T}_a$ are set to the identity matrix. The initial standard deviations for all these parameters are shown in Fig. 2.9.

In this test, we are comparing the performance difference of proposed hybrid EKF algorithm when using points only, vs. when using both point and line features. In both approaches, the estimator is initialized by batch iterative minimization as described in Section 2.7. In Fig. 2.7, the 3D-trajectory estimates of these two approaches are shown. Similar to the first experiment, the cellphone starts and ends at the same physical location. The final position errors of the proposed approach that uses both points and lines is [-0.04 -1.56 0.16] m, which corresponds to 0.17% of the travelled distance, compared to 0.22% when using point features only. By inspection of result from Fig. 2.8, these final errors are commensurate with the uncertainty reported by the estimator, which indicates consistency of the estimator. In addition, Fig. 2.9 shows that the uncertainty of the calibration parameters significantly decreases during the first few seconds. As expected, the reported uncertainties when using both points and lines are smaller than those reported when using points only, as in the simulation.

2.10 Conclusion

In this chapter, we have presented two main contributions. First, we propose a novel parameterization for 3D lines, which is shown to exhibit better linearity properties than alternatives, and thus is better suited for use in linearization-based estimators such as the EKF. Second, we describe a method for processing the observations of lines that is suitable for use with RS sensors, which are found in the majority of low-cost cameras. Our approach forgoes line-fitting, and relies on processing the measurement of each line

pixel individually, thus avoiding assumptions on the shape of the projection of a line in the image. Last but not least, we propose a self-calibration approach that use line feature in addition to traditional point features to calibrate all parameters of a high-fidelity model of both IMU and camera. We demonstrate that the proposed approach is able to process line feature even only coarse estimates of sensors' model parameters are available. These three contributions are employed in conjunction with the hybrid EKF estimator for visual-inertial odometry. The simulation and experimental results we present demonstrate that the proposed approach leads to an improvement in performance, compared to using point features alone, and that the proposed line parameterization outperforms those proposed in prior works.

Algorithm 1 Hybrid EKF algorithm

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

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
  - Acquire feature estimates for both points and lines
  - Computer all feature measurements residuals and associated Jacobians, and then use method in [54] to marginalize feature parameters
  - Perform a Mahalanobis-distance gating test.
- For the features within the state vector, compute the residuals and measurement Jacobians.
- 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 from the state vector features that are no longer tracked.
- Remove all sliding-window states that have no active feature tracks associated with them.

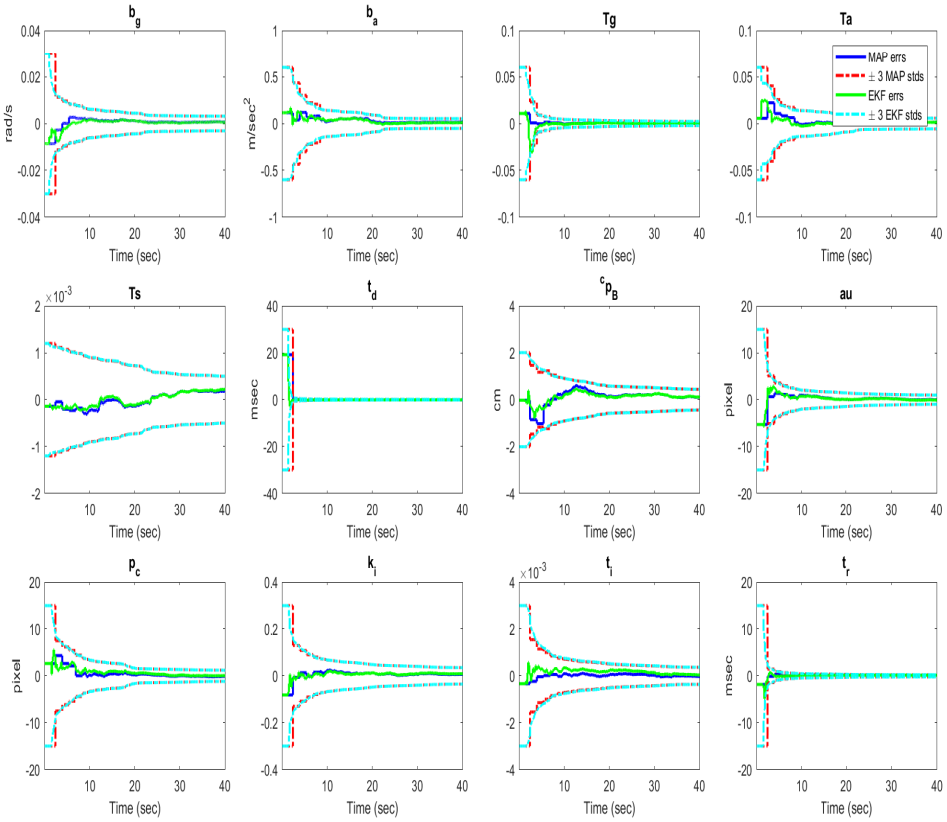

Figure 2.3: Results of a representative simulation trial: estimation errors and the reported ± 3 standard deviations. (Left to right, top to bottom) (a) gyroscope biases, (b) accelerometer biases, (c) gyroscope misalignment/scale, (d) accelerometer misalignment/scale, (e) g-sensitivity, (f) camera-to-IMU 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.

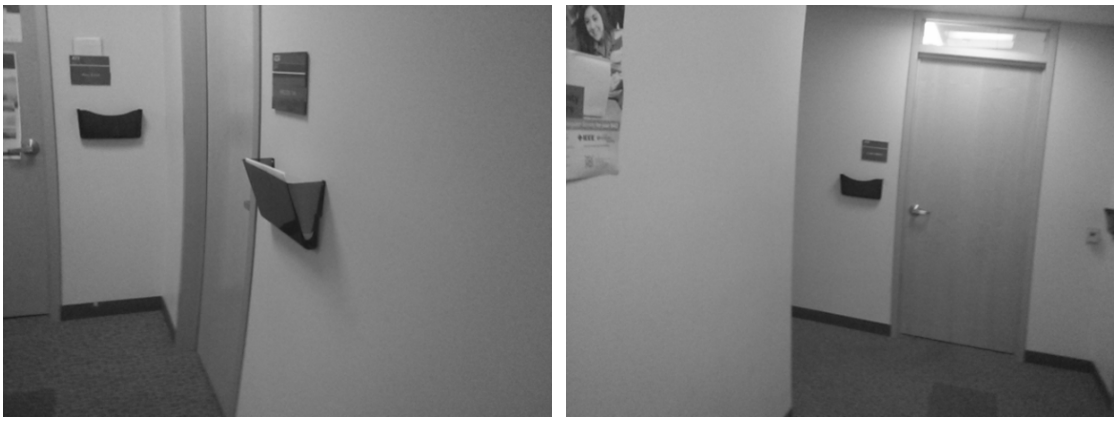

Figure 2.4: Sample images recorded during the experiment in office area.

Figure 2.5: Sample images recorded during the experiment in library.

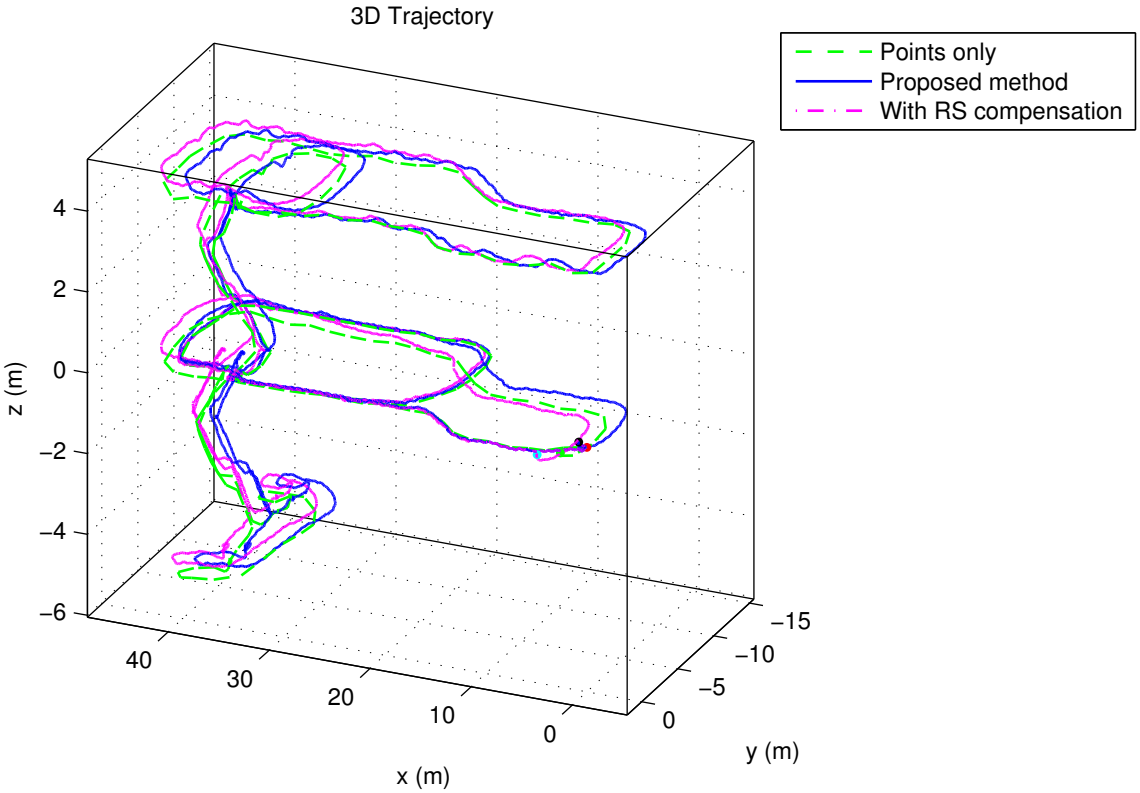

Figure 2.6: Real world experiment: Estimated trajectory when using different settings. The red dot represents the starting point of the trajectory. The black dot represents the endpoint of the trajectory of the Proposed method, the green dot represents the endpoint of the trajectory of Points-only method and the cyan dot represents the endpoint of the trajectory of Lines without RS model method.

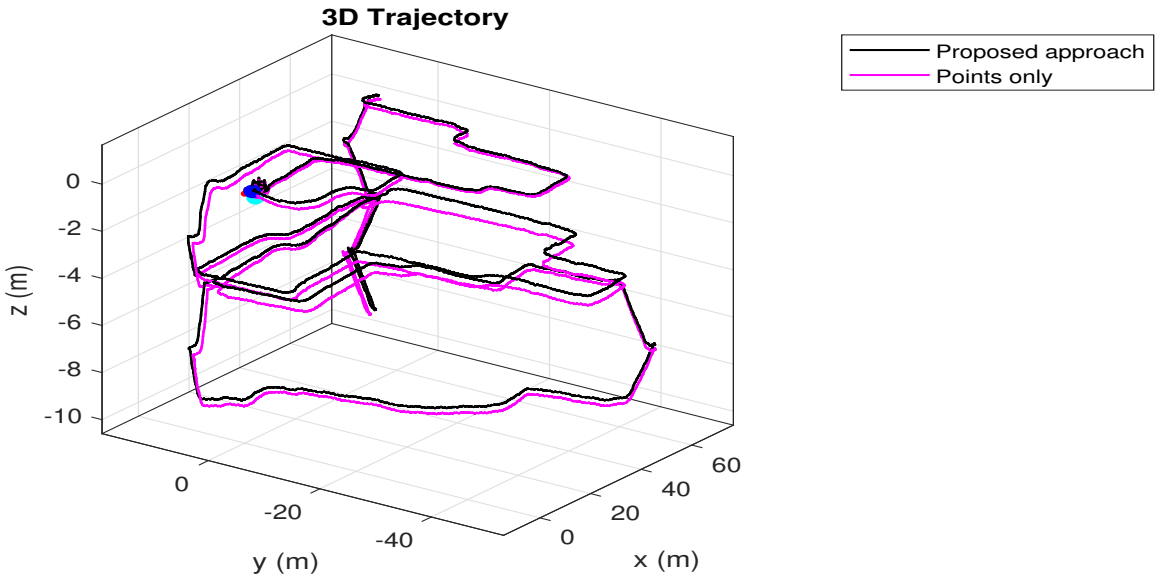

Figure 2.7: Real world experiment: Estimated trajectory when using different settings. The red dot represents the starting point of the trajectory. The blue dot represents the endpoint of the trajectory of proposed approach that uses both point and line features, the cyan dot represents the endpoint of the trajectory of the approach that uses point feature only.

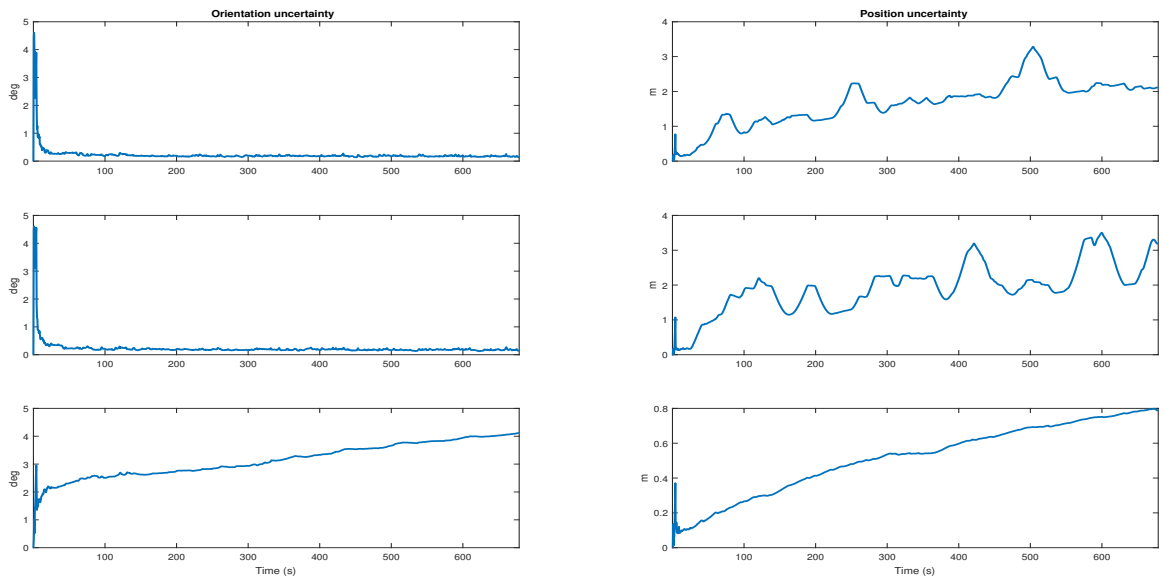

Figure 2.8: Indoor personal-localization experiment: Reported uncertainty. The orientation uncertainty about the $x - y - z$ axes ($roll - pitch - yaw$) is displayed on the left. The figure on the right shows the uncertainty in the $x - y - z$ axis of position. The z axis corresponds to elevation, while the x and y axes to the errors in the horizontal plane.

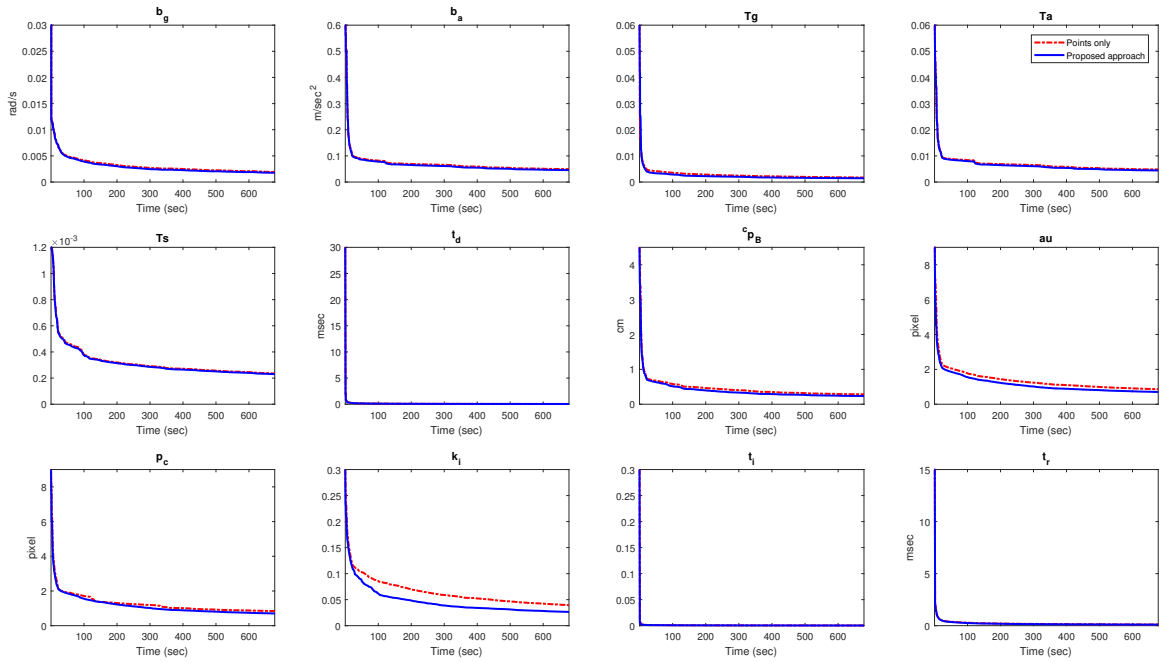

Figure 2.9: Results of real world experiment: the reported ±3 standard deviations. (Left to right, top to bottom) (a) gyroscope biases, (b) accelerometer biases, (c) gyroscope misalignment/scale, (d) accelerometer misalignment/scale, (e) g-sensitivity, (f) camera-to-IMU time offset, (g) camera-to-IMU position (h) camera focal lengths, (i) camera principal point, (j) camera radial distortion, (k) camera tangential distortion, and (l) rolling shutter readout time.

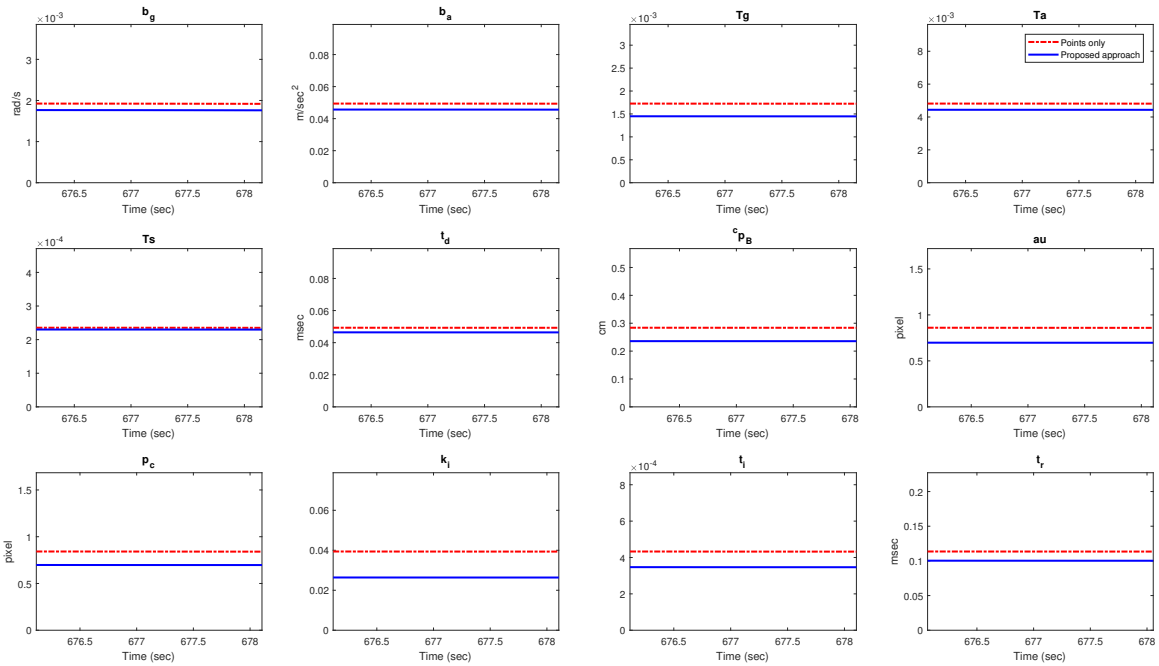

Figure 2.10: Results of real world experiment: the reported ±3 standard deviations, zoomed-in at the end of experiment. (Left to right, top to bottom) (a) gyroscope biases, (b) accelerometer biases, (c) gyroscope misalignment/scale, (d) accelerometer misalignment/scale, (e) g-sensitivity, (f) camera-to-IMU 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.

Chapter 3

Edge-based Visual-Inertial Odometry

3.1 Introduction

The vast majority of algorithms developed for VIO to date rely on the use of *point features* (e.g., Shi-Tomasi corners [64], SIFT features [50], or FAST corners [62], among others), detected and tracked in consecutive images. Using point features greatly reduces the amount of data that needs to be processed (going from hundreds of thousands of pixels to a few hundreds of point features), and therefore makes operation possible even on heavily resource-constrained systems.

However, point-feature extraction also leads to loss of information, as not all parts of the image are used. Therefore, there has been increased interest recently in *direct* methods, which directly use the image intensities as measurements for estimation [17–20, 56].

These methods make it possible to use significantly more pixel locations (theoretically, even the entire image), and thus potentially exploit more information for motion estimation. However, direct approaches also face a number of shortcomings. First, they are very sensitive to errors in the projection geometry (caused, for instance, by inaccurate estimates of the camera intrinsics, or from the errors in the pose estimates themselves) [18]. Thus direct methods may fail in cases where low-cost hardware and/or low-texture environments lead to significant errors in the predicted location of the projection of scene points. Second, the process of image formation in a camera is complex, making it difficult to derive a precise photometric model that considers all possible factors. For example, the image intensity at a given pixel location is affected by the surface properties of scene objects, the viewing angle between the camera and the observed objects, global lighting conditions, camera exposure time, camera gains, and lens characteristics such as vignetting. These factors may be hard to model (e.g., the surface reflectance properties), and may change unpredictably (e.g., lighting conditions), thus leading to unmodeled errors.

To address this problem, direct methods may employ a photometric model in which the intensities of the projection of the same 3D point in two different images, $I_1(r, c)$ and $I_2(r', c')$, are related by a linear expression of the form $I_2(r', c') = \alpha I_1(r, c) + \beta$ [18]. The "gain" and "offset" parameters α, β are good representations of some of the factors mentioned above (e.g., global illumination changes and camera gain/exposure changes), but in practice are also necessary in order to *approximately* model additional effects, such as non-Lambertian surface properties. Estimating these parameters reliably presents challenges (e.g., using the same values for all pixels in an image may not work well), and also increases

the computational cost, as extra states need to be estimated.

Motivated by the above, we here propose a different approach to motion estimation that lies, in some respects, between direct methods and point-feature-based ones. Specifically, in our approach we employ image *contours*, detected by applying edge-detection in an image. In contrast to point-feature methods, we are able to utilize information from *all* parts of an image where gradients with large magnitude exist. This is similar to semi-dense direct methods that use all image areas with significant gradient [17]. However, in contrast to such direct methods our measurement model relies on the geometry of contours in the image, rather than the raw image intensities (in that sense, our method is an *indirect* one). Importantly, the edge contours we employ (defined as the locations where the image gradient magnitude is maximum, along the gradient direction) are *invariant* to linear image intensity changes of the form $\alpha I(r, c) + \beta$. This provides robustness against scene illumination changes and camera gain/exposure changes, and potentially to additional unmodeled effects, if these can be locally well approximated by a linear model.

The use of measurements derived from edge contours provides a number of additional advantages. For instance, edges may be abundant in environments where point features are sparse (e.g., indoors). This observation has also been exploited in previous work, where straight-line features have been used for pose estimation [39, 78]. In contrast to such approaches, however, we do *not* employ any parametric model for the shape of the edges we are using. Instead, each curve in the world is represented as a collection of 3D points, each parameterized by a minimal, two-parameter representation. This allows us to utilize any edge detected in an image, not only those that conform to a certain model, mak-

ing the approach applicable in any environment. It also allows for a simple measurement model to be derived, based on the reprojection errors of the 3D curve points.

In what follows, we present the details of our approach. The proposed contour-based formulation is employed for VIO within the multi-state constraint Kalman filter (MSCKF) [44, 54], which relies on maintaining in the state vector a sliding window of camera poses, while features are directly marginalized and never included in the state. This formulation results in computational complexity that is only linear in the number of edge points. In addition to our novel parameterization of the 3D curves, and the associated measurement model, we here describe a method for computing the accuracy of the edge pixels' location, based on a local bicubic approximation of the image. Our simulation and experimental results show that the resulting formulation outperforms the corresponding point-based approach.

3.2 Related Work

The vast majority of VIO methods to date have relied on the use of point features, which are detected and tracked in the images (see, e.g., [29,32,44] and references therein). In these methods, the measurements are formulated in geometric space (i.e., the coordinates of a feature's projection in the image). By contrast, direct methods for pose estimation formulate a measurement model in the image intensity space, computed for either all image pixels [56], or for image regions with significant gradient magnitude [17–19, 73], or only around a set of extracted feature points [20, 31, 68]. As discussed in the preceding section, the method we propose in this paper seeks to leverage the advantages of both approaches.

By computing measurement residuals in geometric space, we obtain robustness to changes in the lighting and imaging conditions, similarly to point-feature based methods. Moreover, by utilizing image edges, we are able to exploit information from more parts of the image, similarly to semi-dense direct methods.

The use of edge information for *model-based* pose estimation has a long history in computer vision (see, e.g., [3, 23, 38]). In these approaches and their descendants, the 3D structure of the scene (or of an object whose pose is being tracked) is assumed to be known. The 3D camera pose is obtained by minimizing the "reprojection errors" between the observed edges and those predicted based on the known 3D model. In our work, however, we are focusing on estimating trajectory only without prior knowledge of environment. Therefore, knowledge of a 3D model of the scene cannot be assumed, and this type of methods are not applicable.

For edge-based pose estimation in unknown environments, a number of approaches exist that employ a stereo pair of cameras [60, 72] or a depth camera [41]. These approaches are conceptually similar to the model-based ones. Specifically, in both cases, given either a stereo image pair or a depth image, one can create a 3D model for the locations of the edge points in the scene. Then, pose estimation can proceed by finding the transformation that optimally aligns a new image pair or depth scan to that 3D model. These methods rely crucially on having a depth estimate for each of the edge pixels detected in an image (obtained via stereo triangulation or via the depth camera). However, since in our work we employ a monocular camera, this requirement is not met, and the aforementioned approaches cannot be used.

For monocular localization in unknown environments, the VIO algorithms of [39, 78] rely on the use of *straight-line* edges. Since only straight lines can be used, the applicability of these methods is limited to environments where such features are abundant. Moreover, there is a risk of introducing unmodeled errors, when lines that are slightly curved are treated as straight. In a similar vein, the methods of [13, 37, 65] rely on the use of straight-line segments. By contrast, in our work we make no assumptions on the shape of the curves that result in the observed image edges. We also note that higher-order curve parameterizations have been employed in the literature, such as B-splines and NURBS models [57, 77]. While able to handle a larger class of curves than straight lines, these representations are also susceptible to modeling errors, and make the correspondence problem harder, as typically the entire curve must be visible in all images.

The work that is closest to ours is that of Tarrio and Pedre [69]. Similarly to our approach, [69] employs a monocular camera for localization, and a general, point-based parameterization for the edges. However, pose constraints are only computed between pairs of images. By contrast, in our work, *all* the constraints created when an edge is observed in multiple images are utilized, via the use of the MSCKF formulation. Additionally, the algorithm of [69] uses an over-parameterization of edge points, representing each of them as a general 3D point. In our work, a minimal, 2-parameter representation of the edge points is used. Finally, we model the accuracy of the detection of each edge pixel, instead of treating it as a constant. In what follows, we describe the details of our approach.

3.3 Estimator formulation

We begin by describing the formulation of the estimator that we employ for VIO. This is based on the MSCKF algorithm [44, 54], which is an extended-Kalman-filter (EKF)-based method. The state vector of the estimator contains a sliding window of poses, corresponding to the time instants the latest m images were recorded. The IMU readings are used to propagate the IMU state, while the image observations are used to derive probabilistic constraints on the camera poses. We here briefly present the MSCKF estimator, to introduce the notation for the remainder of the paper. The interested reader is referred to [44, 54] for additional details.

Our goal is to estimate the pose of a moving platform with respect to a gravity-aligned global coordinate frame $\{G\}$, using IMU readings and images recorded by a global-shutter camera. To formulate the estimator equations, we affix a coordinate frame $\{I\}$ to the IMU, and a coordinate frame $\{C\}$ to the camera, respectively. We here assume that the intrinsic parameters of the camera, as well as the relative transformation between the camera and the IMU are known via prior calibration. This is done for ease of presentation, and also because these assumptions hold true for our experimental setup. However, these are not requirements for the method we present. If any of the sensor calibration parameters are not accurately known, they can be estimated online, as described in [46].

The state vector of the MSCKF at time-step k is given by:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{I_{k-1}}^T & \cdots & \mathbf{x}_{I_{k-m}}^T \end{bmatrix}^T \tag{3.1}$$

Algorithm 2 Edge-based MSCKF Algorithm

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

Update: When camera measurements become available:

- Augment the sliding window with a new state, and begin image processing.
- For each completed edge-point track
  - Obtain the maximum-likelihood estimate, $\hat{\mathbf{f}}_i$, for the edge point's parameters shown in (3.8), using all the observations of this edge point in a least-squares minimization.
  - Compute the residuals associated with all the edge point measurements, as shown in (3.9), and their Jacobians (equation (3.10)), and apply the method of [54] to remove the term involving $\tilde{\mathbf{f}}_i$.
  - Perform a Mahalanobis-distance gating test.
- Perform an EKF update using all the edge points that pass the Mahalanobis test

State Management:

- Remove all sliding-window states that have no active edge point tracks associated with them, or poses older than m.

where $\mathbf{x}_{E_k}$ is the "evolving state" of the IMU:

$$\mathbf{x}_{E_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_k^T & {}^{G}\mathbf{v}_k^T & \mathbf{b}_{\mathbf{g}_k}^T & \mathbf{b}_{\mathbf{a}_k}^T \end{bmatrix}^T \tag{3.2}$$

The IMU state comprises the unit quaternion ${}^{I_k}_{G}\bar{\mathbf{q}}$, representing the rotation from frame $\{G\}$ to the IMU frame $\{I\}$ at time-step k, the IMU position and velocity in the global frame, ${}^{G}\mathbf{p}_k$ and ${}^{G}\mathbf{v}_k$, respectively, as well as the gyroscope and accelerometer biases, $\mathbf{b}_{\mathbf{g}_k}$ and $\mathbf{b}_{\mathbf{a}_k}$, respectively, which are modeled as Gaussian random-walk processes.

The error-state for the evolving IMU state is defined as [44]:

$$\tilde{\mathbf{x}}_{E_k} = \begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}_k^T & {}^{G}\tilde{\mathbf{p}}_k^T & {}^{G}\tilde{\mathbf{v}}_k^T & \tilde{\mathbf{b}}_{\mathbf{g}_k}^T & \tilde{\mathbf{b}}_{\mathbf{a}_k}^T \end{bmatrix}^T \tag{3.3}$$

where the standard additive error definition is used for the position, velocity and biases (i.e., for a random variable $\mathbf{y}$, its estimate is denoted $\hat{\mathbf{y}}$, and 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 in [44].

Each of the states $\mathbf{x}_{I_j}$, $j = k-m, \ldots, k-1$ comprises the IMU position and orientation at the time instant the corresponding image was recorded:

$$\mathbf{x}_{I_j} = \begin{bmatrix} {}^{I_j}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_j^T \end{bmatrix}^T \quad j = k-m, \ldots, k-1$$

and the errors in the estimates of these states are defined accordingly:

$$\tilde{\mathbf{x}}_{I_j} = \begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}_j^T & {}^{G}\tilde{\mathbf{p}}_j^T \end{bmatrix}^T \quad j = k-m, \ldots k-1$$

When an IMU measurement is received, it is used to propagate the evolving state and the filter covariance matrix, as described in [44]. On the other hand, when a new image is received, the sliding window of states is augmented with a new pose, and the

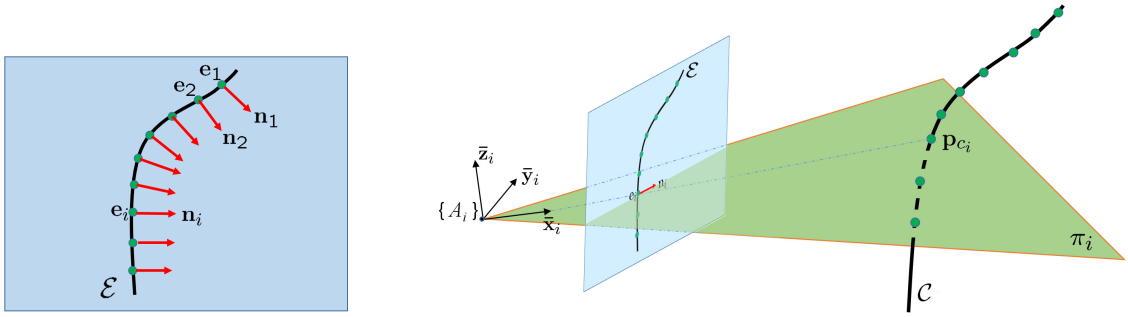

Figure 3.1: Edge Parameterization. Left: Given the first observation of a new curve, we define a number of edge points $\mathbf{e}_i$, spaced at regular intervals along the edge. Right: Each of the edge points, $\mathbf{e}_i$, together with the corresponding edge normal, $\mathbf{n}_i$, and the camera optical center, define a plane π_i. We parameterize the intersection point of π_i with the 3D curve, $\mathbf{p}_{c_i}$, with a 2-parameter vector $\mathbf{f}_i$, representing the position of $\mathbf{p}_{c_i}$ in π_i.

image is processed to extract and match edge points. Each edge point is tracked for as long as possible (or until the maximum limit of m poses is reached), and all its measurements are processed together once the tracking is complete, to provide constraints involving the poses of the sliding window. For this purpose, the multi-state-constraint method of [54] is employed, which makes it possible to use the feature measurements without including the feature in the EKF state vector.

Prior to using an edge point's measurements for an EKF update, a Mahalanobis-distance gating test is performed, to remove outliers. All the edge points that pass the gating test are then employed for an EKF update. At the end of the update, edge points that are no longer visible and old sliding-window states with no active edge-point 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 estimator employs fixed linearization points for each state [44].

3.4 EKF Update with Edges

In this section we present the key contributions of this paper, namely the parameterization of 3D curves that we employ for estimation, and the derivation of the accompanying measurement model for edge observations.

3.4.1 Edge Point Parameterization

In this work, we do not assume any parametric form for the edges in the images, or for the 3D curves whose projection forms these edges. Instead, a 3D curve $\mathcal{C}$ is represented by a set of points, $\mathbf{p}_{c_i}$, $i = 1, \ldots, N$ (the number of points on each curve will generally be different). To define these points, we start with the first observation of the curve. Let $\mathcal{E}$ denote the edge that results from the first observation of $\mathcal{C}$ in an image. We define a number of edge points $\mathbf{e}_i$, $i = 1, \ldots, N$, spaced at regular intervals along $\mathcal{E}$ (see Fig. 3.1). Each of the 3D curve points $\mathbf{p}_{c_i}$ is defined as the intersection of the 3D curve with the plane π_i that contains the edge point $\mathbf{e}_i$, the camera optical center, and the edge normal vector $\mathbf{n}_i$ (see Fig. 3.1).

The motivation for defining the set of points $\mathbf{p}_{c_i}$ as described above, i.e., via the intersection of $\mathcal{C}$ with a set of planes π_i, is that this leads to a *two-dimensional parameterization* for each of the points $\mathbf{p}_{c_i}$. These two parameters describe the position of the intersection of the curve $\mathcal{C}$ with the *known* two-dimensional plane π_i. This is a desirable property: note that changes in the position of $\mathbf{p}_{c_i}$ along the curve are *unobservable*, since, in general, we cannot distinguish different points along an edge in an image. Therefore, if the point $\mathbf{p}_{c_i}$ was represented by three parameters (the standard representation for points

in 3D), this would be an over-parameterization, which is not suitable for the problem at hand.

We now describe the definition of the two parameters that we use to represent each of the points $\mathbf{p}_{c_i}$, $i = 1, \ldots, N$. Given the normalized image coordinates of an edge point, $\mathbf{e}_i$, and the edge-normal vector at this point (i.e., the image gradient vector), $\mathbf{n}_i$, we define a *known, constant* coordinate frame $\{A_i\}$, whose origin coincides with the estimate of the camera coordinate frame, and whose coordinate axes are defined in the global frame as:

$$^{G}\bar{\mathbf{x}}_i = \frac{1}{\eta_x} {}^{G}_{C}\hat{\mathbf{R}} \begin{bmatrix} \mathbf{e}_i \\ 1 \end{bmatrix} \tag{3.4}$$

$$^{G}\bar{\mathbf{z}}_i = \frac{1}{\eta_z} {}^{G}_{C}\hat{\mathbf{R}} \left(\begin{bmatrix} \mathbf{e}_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{n}_i \\ 0 \end{bmatrix} \right) \tag{3.5}$$

$$^{G}\bar{\mathbf{y}}_i = {}^{G}\bar{\mathbf{z}}_i \times {}^{G}\bar{\mathbf{x}}_i \tag{3.6}$$

where ${}^{G}_{C}\hat{\mathbf{R}}$ is the estimate for the rotation matrix between the camera frame and the global frame, and η_x and η_z are normalization constants to ensure unit length of the corresponding vectors. Intuitively, the above definitions mean that the x-axis of the frame $\{A_i\}$ is along the vector from the camera optical center to $\mathbf{e}_i$, and the z-axis is normal to the plane π_i. With this definition of $\{A_i\}$, the point $\mathbf{p}_{c_i}$ has a z-coordinate of zero, and we therefore parameterize it as:

$$^{A_i}\mathbf{p}_{c_i} = \frac{1}{\rho_i} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{bmatrix} \tag{3.7}$$

In the above parameterization, ρ_i has the role of inverse depth, which is known to lead to improved linearity properties in vision-based estimation [53]. Moreover, we note that with our frame definition, the initial estimate of the parameter θ_i is by definition zero.

To summarize, given the image edge corresponding to a new 3D curve, we define a set of points along the image edge, $\mathbf{e}_i$, $i = 1, \ldots, N$, and subsequently we define a set of *known, constant* coordinate frames $\{A_i\}$, whose origin coincides with the origin of the estimated camera frame, and their principal axes are defined as shown in (3.4)-(3.6). We subsequently parameterize the intersections of the 3D curve with the $x - y$ planes of these frames by the two-parameter vector:

$$\mathbf{f}_i = \begin{bmatrix} \rho_i & \theta_i \end{bmatrix}, \quad i = 1, \ldots, N \tag{3.8}$$

which defines the points according to (3.7).

3.4.2 Measurement model

To derive a measurement model based on the parameterization described in the preceding section, we rely on the fact that the projections of all points $\mathbf{p}_{c_i}$, $i = 1, \ldots, N$ of a 3D curve, should lie on the edges corresponding to the curve in all images. Therefore, if we denote by $\mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i)$ the predicted projection of $\mathbf{p}_{c_i}$ on the image at time step j, the following quantity can be viewed as a measurement residual:

$$r_{d_{ij}} = \text{dist}(\mathcal{E}_j, \mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i))$$

where $\mathcal{E}_j$ denotes the observed edge resulting from the projection of the curve on the image at time step j, while $\hat{\mathbf{x}}_{I_j}$ and $\hat{\mathbf{f}}_i$ are the estimates for the IMU pose and the curve-points'

parameters, used to predict the projection of $\mathbf{p}_{c_i}$ on the image (the exact form of the projection function, $\mathbf{h}$, is shown in Section 3.4.3). While valid, the above expression is not suitable for use in an EKF estimator, as it is a nonnegative quantity. To address this issue, we instead define the following residual, which is a signed quantity:

$$r_{ij} = \mathbf{n}_{ij}^T \left(\mathbf{z}_{ij} - \mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i) \right) \tag{3.9}$$

where $\mathbf{z}_{ij}$ is the point on $\mathcal{E}_j$ that is closest to $\mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i)$, and $\mathbf{n}_{ij}^T$ is a constant unit vector. Note that, for any unit vector, the above quantity would be a valid residual, in the sense that if the estimates were perfect and the edge measurement noiseless, it would equal zero. However, to obtain a meaningful residual, we choose the vector $\mathbf{n}_{ij}$ as the gradient of the image at $\mathbf{z}_{ij}$. This is illustrated in Fig. 3.2. It is important to point out that the residual in (3.9) is a *scalar* quantity, which is desirable, since edge observations can only provide useful information in the direction normal to the gradient.

To obtain a linearized approximation of the residual, we begin by using first-order Taylor expansion of $\mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i)$, which yields:

$$\mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i) \simeq \mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i) + \mathbf{H}_{\mathbf{x}_{ij}} \tilde{\mathbf{x}}_{I_j} + \mathbf{H}_{\mathbf{f}_{ij}} \tilde{\mathbf{f}}_i$$

where $\tilde{\mathbf{x}}_{I_j}$ and $\tilde{\mathbf{f}}_i$ are the errors in the estimates for the IMU pose at time step j and for the point's parameters, respectively, and $\mathbf{H}_{\mathbf{x}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ are the corresponding Jacobians. Moreover, let $\mathbf{z}_{ij} = \check{\mathbf{z}}_{ij} + \boldsymbol{\eta}_{ij}$, where $\check{\mathbf{z}}_{ij}$ is the true location of the point on the edge, and $\boldsymbol{\eta}_{ij}$ the noise in its detection. Substituting in (3.9), and noting that $\check{\mathbf{z}}_{ij} = \mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i)$, we obtain the following linearized form of the residual:

$$r_{ij} \simeq \mathbf{H}_{\mathbf{x}_{ij}} \tilde{\mathbf{x}}_{I_j} + \mathbf{H}_{\mathbf{f}_{ij}} \tilde{\mathbf{f}}_i - \mathbf{n}_{ij}^T \boldsymbol{\eta}_{ij} \tag{3.10}$$

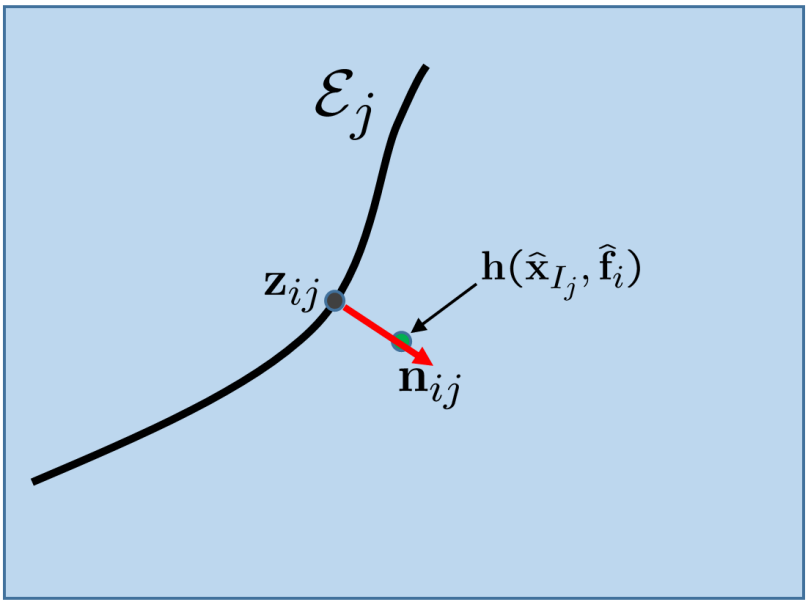

Figure 3.2: Illustration of the measurement model.

It is important to note that the term $\mathbf{n}_{ij}^T \boldsymbol{\eta}_{ij}$ represents the noise in the detection of the point that is closest to $\mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{f}}_i)$, *along the direction of the gradient*. In other words, this noise term corresponds to the accuracy with which the edge can be detected in the image. In Section 3.4.4, we describe how the covariance of this noise term can be computed.

Once the residuals defined in (3.9), as well as their corresponding Jacobians in (3.10), have been computed for all the images in which the curve point can be tracked, the method described in [54] is employed in order to marginalize out the parameter error $\mathbf{f}_i$, and a Mahalabobis gating test is performed. If the test is successful, the measurements of this curve point are used for an EKF update, along with all other curve points available at this time, as detailed in [54].

3.4.3 Geometric Model

We now present the function $\mathbf{h}$, which describes the projection of a curve point $\mathbf{p}_{c_i}$ with parameter vector $\mathbf{f}_i$, on the image at time step j. We begin by computing the position of $\mathbf{p}_{c_i}$ with respect to the camera at time step j:

$$^{C_j}\mathbf{p}_i = {}^{C}_{I}\mathbf{R}\,{}^{I_j}_{G}\mathbf{R}\left({}^{G}\mathbf{p}_{A_i} + {}^{G}_{A_i}\mathbf{R}\,{}^{A_i}\mathbf{p}_{c_i} - {}^{G}\mathbf{p}_{I_j}\right) + {}^{C}\mathbf{p}_I$$

where ${}^{C}_{I}\mathbf{R}$ and ${}^{C}\mathbf{p}_I$ are the known rotation and translation between the camera and IMU frames, ${}^{I_j}_{G}\mathbf{R}$ and ${}^{G}\mathbf{p}_{I_j}$ describe the IMU pose in the global frame, ${}^{G}_{A_i}\mathbf{R}$ and ${}^{G}\mathbf{p}_{A_i}$ are the known, constant parameters of the frame $\{A_i\}$ associated with the point $\mathbf{p}_{c_i}$, and ${}^{A_i}\mathbf{p}_{c_i}$ is given by (3.7).

Given the vector ${}^{C_j}\mathbf{p}_i = [{}^{C_j}x_i \;\; {}^{C_j}y_i \;\; {}^{C_j}z_i]^T$, the image coordinates of its projection on image j depend only on the imaging geometry of the camera. In our experiments, we employ a fisheye-lens camera, and we use the model of [12] to describe the projection geometry:

$$\mathbf{h}\left(\mathbf{x}_{I_j}, \mathbf{f}_i\right) = \frac{1}{r_u\omega}\arctan\left(2r_u\tan\left(\frac{\omega}{2}\right)\right)\begin{bmatrix} a_u u \\ a_v v \end{bmatrix} + \mathbf{p}_p \tag{3.11}$$

where $\mathbf{p}_p$ 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

$$r_u = \sqrt{u^2 + v^2} \tag{3.12}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^{C_j}z_i}\begin{bmatrix} {}^{C_j}x_i \\ {}^{C_j}y_i \end{bmatrix} \tag{3.13}$$

All the camera intrinsic parameters, namely $\mathbf{p}_p, a_u, a_v$, and ω, are known through prior calibration.

3.4.4 Edge Detection And Uncertainty Model

We now describe our approach for detecting edges in the image with subpixel precision, and estimating the accuracy with which they are localized. Our approach is based on the method of [24], where a bicubic polynomial is employed to model the local image intensity. In particular, edges are first detected to pixel-level accuracy by applying a Canny-like edge detection method on the images [8]. Then, for those pixels where subpixel accuracy is required (i.e., those used to define the curve points as described in Section 3.4 and their correspondences) we fit the following bicubic polynomial to model the intensity of the image in a local 7×7 pixel neighborhood:

$$\begin{aligned} f\,(c,\,r) = k_1 + k_2 c + k_3 r + k_4 c^2 + k_5 cr + k_6 r^2 \\ + k_7 c^3 + k_8 c^2 r + k_9 c r^2 + k_{10} r^3 \end{aligned} \tag{3.14}$$

where c and r are the column and row coordinates with respect to the center edge pixel. We then obtain the sub-pixel location of the edge, as well as its accuracy, based on the parameters of this polynomial. Specifically, note that the direction of the image gradient at the edge pixel under consideration is given by $[k_2 \;\; k_3]^T$. The location of the edge is the point, along this direction, where the second-order directional derivative along the gradient is zero. By solving the resulting set of equations, we obtain the following subpixel offset for the location for the edge, along the direction of the gradient:

$$o(\mathbf{k}) = \frac{(-k_2^2 k_4 - k_2 k_3 k_5 - k_3^2 k_6)\sqrt{k_2^2 + k_3^2}}{3(k_2^3 k_7 + k_2^2 k_3^2 + k_2 k_3^3 k_9 + k_3^3 k_{10})} \tag{3.15}$$

where we have made explicit the fact that this offset is a function of the parameter vector $\mathbf{k} = [k_1 \;\; k_2 \;\; \ldots \;\; k_{10}]^T$. As explained in the preceding section, the accuracy with which this offset

can be determined represents the "measurement noise" in our measurement model (3.10). To compute the variance of this noise, we employ linearization to compute the error in $o(\mathbf{k})$ as:

$$\tilde{o}(\mathbf{k}) \simeq \frac{\partial o}{\partial \mathbf{k}} \frac{\partial \mathbf{k}}{\partial \mathbf{I}} \tilde{\mathbf{I}}$$

where $\mathbf{I}$ represents the values of the image intensity in the local 7×7 image neighborhood, and $\tilde{\mathbf{I}}$ the corresponding image-intensity noise. Therefore, the variance of the measurement noise is given by:

$$\sigma_o^2 = \frac{\partial o}{\partial \mathbf{k}} \mathbf{C_k} \frac{\partial o}{\partial \mathbf{k}}^T \tag{3.16}$$

where $\mathbf{C_k}$ is the covariance matrix of the errors in the estimate of $\mathbf{k}$ that we obtain from fitting the model (3.14). This matrix is a function of the image intensity noise variance, but, importantly, it does not depend on the value of $\mathbf{k}$ itself, and can thus be precomputed. By contrast, σ_o^2 will depend on the local image appearance, which affects the Jacobian $\frac{\partial o}{\partial \mathbf{k}}$. This allows us to model the fact that sharper edges can be localized more accurately than smoother ones.

We note that the process for computing the noise variance described above models the effects of image noise, but does not account for additional sources of error. For example, the true image intensity in the local region will, in general, not follow a bicubic-polynomial model. To account for these additional effects, in our implementation we use a threshold on the minimum allowable value of σ_o. Specifically, the value of σ_o that we use is the maximum of the value computed by (3.16), or 0.5 pixels.

3.4.5 Edge point tracking

To perform edge tracking across images we have tested two different approaches: 1) frame-to-frame constraint tracking (termed FFC in the following), and 2) multi-frame constraint tracking (termed MFC). In the first case, matching from image j to image $j+1$ is performed using search along epipolar lines, computed using the predicted relative pose between the two camera frames. For a given edge point in image j, if an edge point is found along the epipolar line in image $j+1$, and the directions of the gradient vectors are close (within a threshold of 30°), then normalized cross-correlation is used as the matching criterion. Once matching for all points from image j is complete, new points are introduced in the edge segments of image $j+1$ where no matches have been detected.

The MFC approach relies on the minimization of a matching cost, computed over multiple frames. This cost function is defined as:

$$\mathbf{c} = \min_{\rho} \sum_{i=2}^{N} \mathbf{d}\left(\rho,\ {}^{C_i}_{C_1}\hat{\mathbf{R}},\ {}^{C_i}_{C_1}\hat{\mathbf{p}}\right) \tag{3.17}$$

where ρ is the inverse depth of an edge point with respect to camera frame $\{C_1\}$, and $\mathbf{d}\left(\rho,\ {}^{C_i}_{C_1}\hat{\mathbf{R}},\ {}^{C_i}_{C_1}\hat{\mathbf{p}}\right)$ is the distance of this point to the closest edge in image i. This distance is computed efficiently by use of the distance transform. To reduce the number of false matches, the edge points in each image are clustered according to their orientation, and a separate distance transform is computed for each set.

When a new edge-point track is initialized, we start with the latest image, and perform the above minimization to find correspondences in the previous $N-1$ images. This minimization takes place using a discrete search (note that a continuous inverse-depth parameter is estimated and used in the filter). To help avoid spurious matches, we reject

matches when the resulting minimum cost is above certain threshold. Additionally we perform a normalized cross-correlation test of all the related intensity patches. A similar process is followed to extend the edge point track into a newly received image. To this end, we project the edge point using the previous depth estimate into the new image and compute the corresponding cost given by the distance transform. If the cost remains below the threshold, and the point also passes the normalized cross-correlation test, we accept this match.

3.5 Experimental results

To test the performance of the proposed method, we conducted a set of Monte-Carlo simulations to evaluate its consistency, as well as a set of real-world experiments to evaluate its accuracy in real-world settings. For the edge-based VIO approach, we define the 3D curve points via edge points spaced every 3 pixels along the first observation of a curve, as described in Section 3.4. These points are detected at subpixel accuracy, using the method in Section 3.4.4.

3.5.1 Monte-Carlo simulations

For the simulations, the trajectory of a hand-held device in a room-sized environment, recorded in a prior experiment, is used to generate a realistic ground-truth motion. Based on this ground truth trajectory, inertial measurements are subsequently generated, with random noise realizations used in each Monte-Carlo trial. To generate simulated camera images, we texture-map images recorded during real-world experiments onto the four

Figure 3.3: Sample images generated by the simulator.

simulated walls of the room, as well as the floor and ceiling. The simulated camera images are created with a different realization of Gaussian noise added to the image intensities in each trial. Two sample images generated in the simulation are shown in Fig. 3.3. The IMU measurements are available at 200 Hz, and the camera images at 10 Hz. The trajectory is approximately 181 m long, traversed in about 2.9 mins.

To evaluate the accuracy and consistency of the edge-based VIO approaches using different tracking methods in comparison to the point-based one, we perform 50 Monte-Carlo simulations. For both formulations, we compute the RMS error for the IMU orientation and position, as well as the normalized estimation error squared (NEES) of the IMU pose at each time step. The RMS errors provide us with a measure of accuracy, while the NEES is a measure of consistency [44]. Specifically, if the estimator is consistent (i.e., if the estimation errors are zero mean, and have ensemble covariance equal to the covariance reported by the filter [4]) the average value of the NEES over all Monte Carlo trials should equal six

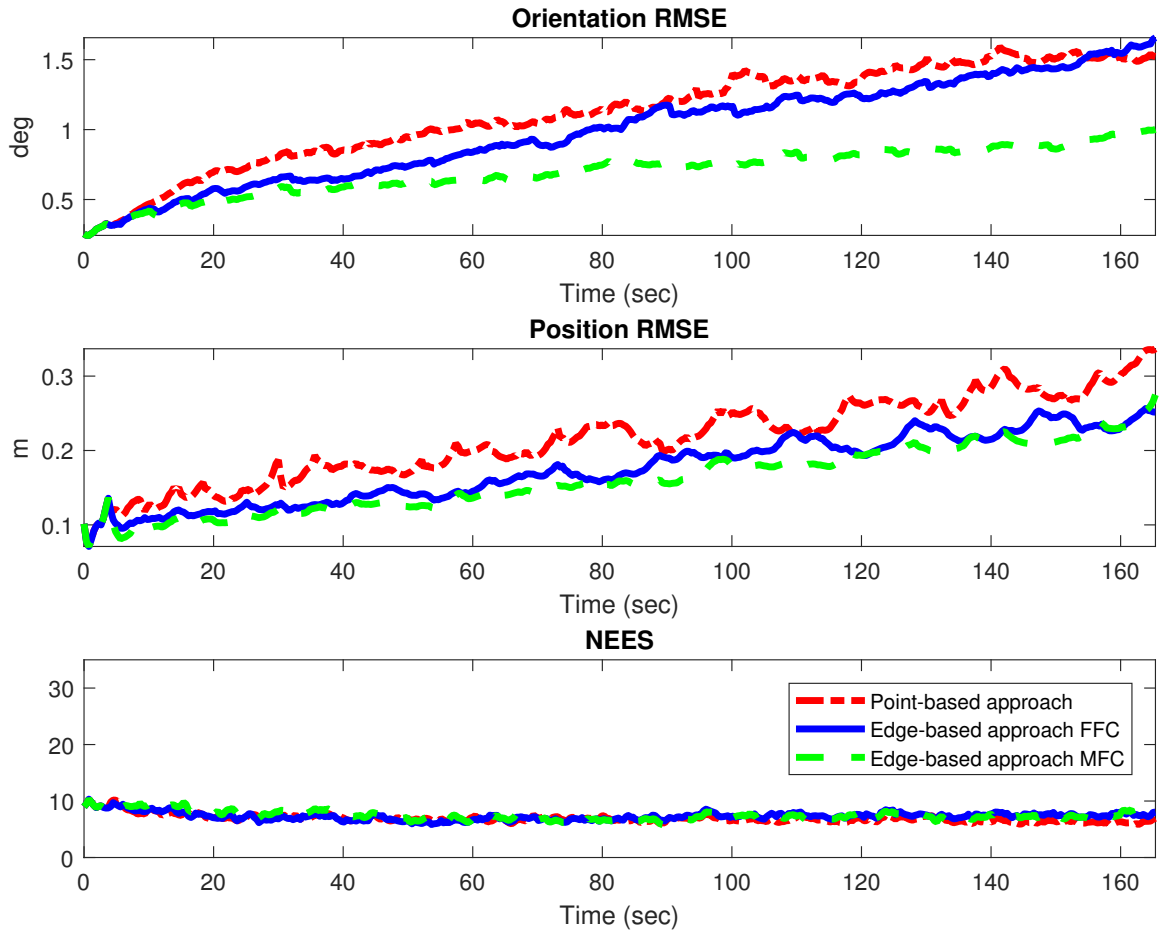

Figure 3.4: Average RMS error and NEES over 50 Monte-Carlo trials.

(i.e., should equal the dimension of the pose-error vector). The average RMS and NEES over all 50 Monte Carlo trials are shown in Fig. 3.4, plotted over time. We observe that the use of MFC tracking yields more accurate results than the FFC approach, and both outperform the point-based formulation in terms of estimation precision. In addition, we note that the NEES of the edge-based approach for using FFC and MFC tracking are 7.36 and 7.69 respectively, close to the ideal value of 6 for consistency.

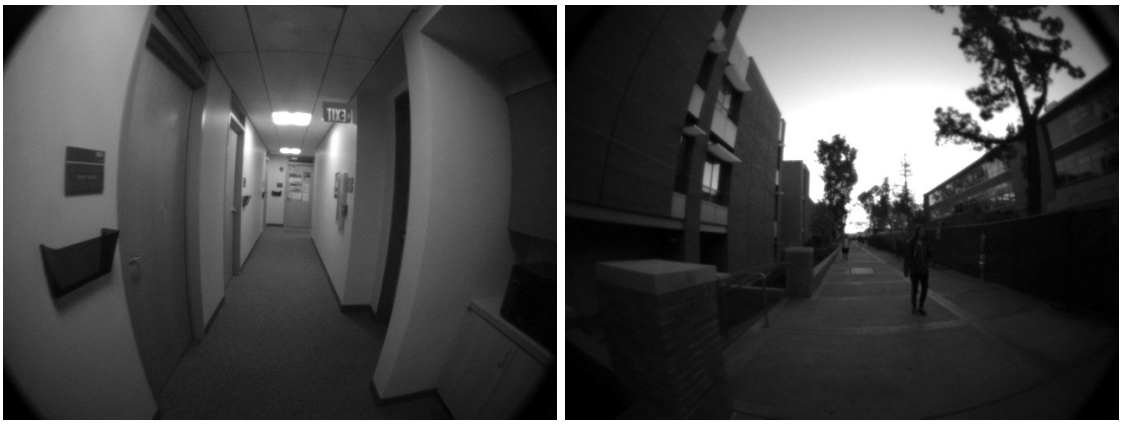

Figure 3.5: Sample images recorded during the experiments.

3.5.2 Real-World Experiments

We next present results from real-world experiments that were conducted to compare the performance of the proposed edge-based approach to the original, point-based formulation of the MSCKF in real-world settings. In the proposed edge-based approach, point features extracted out of image are *not* used. We collected six datasets in both indoor and outdoor environments, with trajectory lengths ranging from 314 m to 465 m (sample images from the experiments are shown in Fig. 3.5). For data collection a Tango developer platform was used, which records IMU data at a sample rate of 200 Hz and images at 30 Hz. All datasets were processed offline, so that comparisons between the different approaches could be performed.

We evaluate the performance of the proposed edge-based MSCKF formulation, against the point-based one, using the performance metrics of [22]. Specifically, we compute the error in the position and orientation estimates of each method as a function of

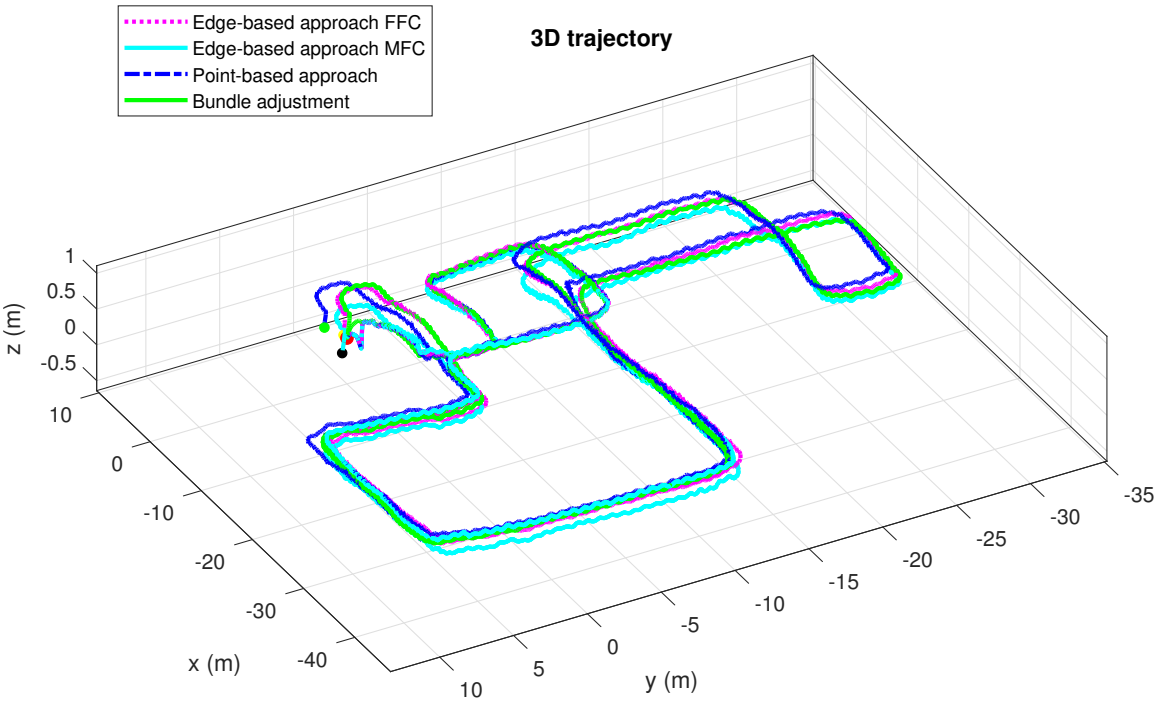

Figure 3.6: Trajectory estimates computed using the edge-based and point-based MSCKF, as well as the bundle-adjustment ground truth, in one of the datasets.

path length. For this to be feasible, the ground truth of the trajectory is required. Since an external ground-truth system is not available in the mixed indoor-outdoor settings where the datasets were collected, we here employ global visual-inertial bundle adjustment to obtain a trajectory estimate that is used as ground truth. This bundle adjustment utilizes point features as the image measurements, and makes use of loop-closure constraints. While collecting data, care was taken so that a significant number of "loop closure events" occur along the trajectories. This ensures that the accuracy of the bundle adjustment is significantly higher than the accuracy of the point- and edge-based VIO, and using it as a ground truth is a reasonable approximation. The trajectory estimates produced by the three approaches in one of the datasets are shown in Fig. 3.6.

The evaluation of the accuracy of the point-based and edge-based approaches is shown in Fig. 3.7. To compute the error for a path length L, the errors in the relative-position and relative-orientation estimates of each method are computed and averaged over all trajectory segments of length L in all datasets. The errors are then divided by L, to obtain the position error as a percentage of the traveled distance, and the orientation-error growth in degrees per meter traveled:

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

where the average is taken over all time intervals $[t_i, t_{i+L}]$ corresponding to path segments of length L, and ${}^{I_{t_i}}\tilde{\boldsymbol{\theta}}_{I_{t_{i+L}}}$ represents the error of the estimate of the relative orientation ${}^{I_{t_i}}_{I_{t_{i+L}}}\mathbf{R}$. The results of Fig. 3.7 show that, due to the use of more information in the images, the proposed edge-based approach outperforms the point-based one, both in terms of position and orientation errors. The average reduction in error for the edge-based method using FFC, over all path lengths, is 41.6% for the orientation, and 26.1% for the position. The MFC edge tracking approach leads to additional improvement in performance, with 43.6% lower errors for the orientation and 44.3% for the position.

3.6 Conclusion

In this chapter, we have presented a novel, edge-based algorithm for monocular visual-inertial odometry. The proposed method relies on a point-based parameterization of 3D curves in which each point is parameterized by only two parameters, defining the

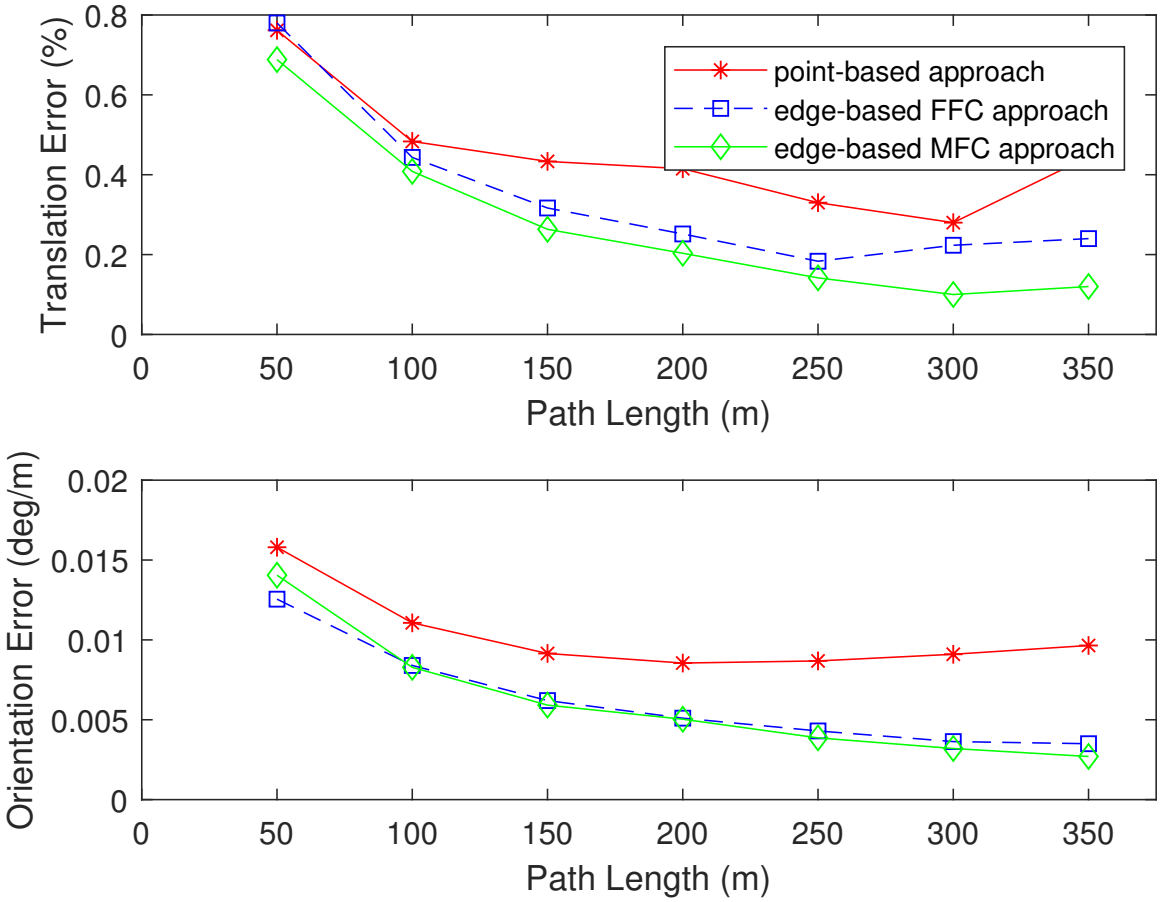

Figure 3.7: Performance comparison of the edge-based MSCKF to the original point-based one. The apparent lower accuracy of the methods over small path lengths is attributed to errors in the estimate of the bundle adjustment that is used as ground truth.

intersection of a known plane with the curve. This minimal parameterization, in conjunction with a measurement model that only employs measurement residuals along the direction of the edge gradient, results in an approach that can use all image edges, without any assumptions on scene geometry. Through extensive experimental validation, both in a simulated 3D environment, as well as in real-world experiments in both indoor and outdoor settings, we have shown that the proposed approach is consistent, and leads to improved

estimation accuracy, compared to the “traditional” point-based one. In addition the use of multi-frame constraint tracking approach leads to additional improvement, due to tighter restrictions on edge point tracking.

Chapter 4

Conclusions

Traditional VIO approaches rely on detection and tracking of feature points in images to reduce the amount of image data to be processed. However, this leads to loss of information for motion estimation, and we have therefore focused on approaches that use more image regions for motion estimation. Specifically, we first examined the use of straight-line features, and demonstrated how they can be used in the hybrid EKF estimator formulator for both GS and RS cameras. We showed that the additional information provided by line features not only improves localization accuracy, but can also lead to better calibration of the sensor models parameters. In addition we proposed a novel, edge-based algorithm for monocular visual-inertial odometry. This approach enables the use of all image areas with significant gradient, without assuming the presence of straight lines in the scene. The results presented in this thesis demonstrate that the use of more general features, in addition to traditional point features, can lead to improved precision in visual-inertial odometry, and holds promise for real-time applications.

Bibliography

[1] Omar Ait Aider, Adrien Bartoli, and Nicolas Andreff. Kinematics from lines in a single rolling shutter image. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 – 6, Minneapolis, MN, Jun 2007.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, June 2012.

[3] M. Armstrong and A. Zisserman. Robust object tracking. In *Proceedings of the Asian Conference on Computer Vision*, pages 58–61, 1995.

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

[5] Adrien Bartoli and Peter Sturm. The 3D line motion matrix and alignment of line reconstructions. *International Journal of Computer Vision*, 57(3):159–178, 2004.

[6] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.

[7] Pierre-Jean Bristeau, Francois Callou, David Vissie, 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.

[8] J.F. Canny. A computational approach to edge detection. *IEEE Transacions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[9] Manmohan Chandraker, Jongwoo Lim, and David Kriegman. Moving in stereo: Efficient structure and motion using lines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1741 – 1748, Kyoto, Japan, October 2009.

[10] 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.

[11] Frank Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1203, Dec. 2006.

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

[13] Ethan Eade and Tom Drummond. Edge landmarks in monocular SLAM. In *Proceedings of the British Machine Vision Conference*, pages 2.1–2.10, Edinburgh, Sep 2006.

[14] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. High-accuracy preintegration for visual-inertial navigation. Technical report, 2016.

[15] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Direct visual-inertial navigation with analytical preintegration. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1429–1435, 2017.

[16] Ali Elqursh and Ahmed M. Elgammal. Line-based relative pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3049–3056, Providence, RI, June 2011.

[17] 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*, pages 1449–1456, 1-8 Dec. 2013.

[18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *CoRR*, abs/1607.02565, 2016.

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

[20] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 31-June 7 2014.

[21] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems*, 2015.

[22] 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)*, pages 3354–3361, Washington, DC, USA, 2012.

[23] D. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, 7(3):243–270, 1992.

[24] Robert M. Haralick. Digital step edge from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:58–68, 1984.

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

[26] 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.

[27] 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.

[28] 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.

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

[30] H. Jin, P. Favaro, and R. Cipolla. Visual tracking in the presence of motion blur. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 18–25 vol. 2, June 2005.

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

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

[33] Neel Joshi, Sing Bing Kang, C. Lawrence Zitnick, and Richard Szeliski. Image deblurring using inertial measurement sensors. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 30:1–30:9, New York, NY, USA, 2010. ACM.

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

[35] 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.

[36] 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.

[37] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Marseille, France, 2008.

[38] D. Koller, K. Danilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, June 1993.

[39] 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.

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

[41] M. Kuse and Shaojie Shen. Robust camera motion estimation using direct edge alignment and sub-gradient method. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 573–579, 16-21 May 2016.

[42] 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.

[43] 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.

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

[45] M. Li. and A. I. Mourikis. Vision-aided inertial navigation with rolling-shutter camera. *International Journal of Robotic Research*, 33(11):1490–1507, 2014.

[46] M. Li, H. Yu, X. 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 (ICRA)*, pages 409–416, May 31-June 7 2014.

[47] 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.

[48] 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.

[49] Yonggen Ling, Tianbo Liu, and Shaojie Shen. Aggressive quadrotor flight using dense visual-inertial fusion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1499–1506, May 2016.

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

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

[52] 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.

[53] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems*, pages 81–88, Philadelphia, PA, Aug. 2006.

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

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

[56] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision*, pages 2320–2327, 6-13 Nov. 2011.

[57] I. Nurutdinova and A. Fitzgibbon. Towards pointless structure from motion: 3D reconstruction and camera parameters from general 3D curves. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2363–2371, 7-13 Dec. 2015.

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

[59] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *2009 IEEE International Conference on Robotics and Automation*, pages 2250–2257, May 2009.

[60] A. Richardson and E. Olson. PAS: visual odometry with perspective alignment search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1053–1059, Chicago, IL, Sept 2014.

[61] K. Roberts. A new representation for a line. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 635 – 640, Ann Arbor, MI, Jun 1988.

[62] 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.

[63] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[64] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, June 1994.

[65] Paul Smith, Ian Reid, and Andrew Davison. Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference*, pages 3.1–3.10, Edinburgh, Sep 2006.

[66] Joan Sola, Teresa Vidal-Calleja, Javier Civera, and Jose Maria Martinez Montiel. Impact of landmark parameterization on monocular EKF-SLAM with points and lines. *International Journal of Computer Vision*, 97(3):339–368, 2012.

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

[68] 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, Sept. 28 2015-Oct. 2 2015.

[69] J. J. Tarrio and S. Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 702–710, 7-13 Dec. 2015.

[70] Camillo J. Taylor and David J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transacions on Pattern Analysis and Machine Intelligence*, 17(11):1021 – 1032, 1995.

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

[72] M. Tomono. Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4306–4311, Kobe, Japan, May 2009.

[73] V. Usenko, J. Engel, J. Stkler, 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, Sweden, May 2016.

[74] 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.

[75] O. Whyte, J. Sivic, and A. Zisserman. Deblurring shaken and partially saturated images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 745–752, Nov 2011.

[76] J. Witt and U. Weltin. Robust stereo visual odometry using iterative closest multiple lines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4164 – 4171, Tokyo, Nov 2013.

[77] Yi Jun Xiao and Y. F. Li. Optimized stereo reconstruction of free-form space curves based on a nonuniform rational B-spline model. *Journal of the Optical Society of America A*, 22(9):1746–1762, Sep 2005.

[78] Hongsheng 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, Sept. 28 2015-Oct. 2 2015.

[79] 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.

[80] Lilian Zhang and Reinhard Koch. Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation*, 25(5):904–915, 2014.