Universität des Saarlandes
Max-Planck-Institut für Informatik

Keyframe-based Visual-Inertial Odometry for Small Workspace

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

Xi Li

angefertigt unter der Leitung von / supervised by
Dr. Roland Angst

betreut von / advised by
Dr. Roland Angst

begutachtet von / reviewers
Dr. Roland Angst
Prof. Dr. Antonio Krüger

Saarbrücken, June 2016

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

Statement in Lieu of an Oath

I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

_____ _____

(Datum / Date) (Unterschrift / Signature)

Acknowledgments

I would like to thank Dr. Roland Angst who has been a great supervisor during my thesis. Thanks Dr. Angst for giving me many valuable advise, also keeping a perfect balance of guidance and freedom.

Also many thanks to members in our group: Dushyant and Rui, who has helped and encouraged me a lot. Special thanks to Han and Jianan, who has suggested a lot of amazing ideas and helped me to debug my code. Thanks Siwen, Hang for being the proof readers of my master thesis.

At last I want to thank my parents, who has always give their unconditional supports and love to me. Without their supports and love, I never would have made it.

Keyframe-based Visual-Inertial Odometry for Small Workspace

by

Xi Li

Submitted to the Department of Computer Science
on June 1, 2016, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

Mobile robot navigation has been an attractive topic in the Robotics area for a long time. *Visual SLAM* simultaneously locates robots' poses and constructs a map via analysing measurements obtained from visual sensors. However due to the costly computational complexity of camera, it is non-trivial to gain a high-quality pose estimation while the whole system runs in real-time. *Inertial Measurement Unit* (IMU) measures the rotational rate and acceleration of the robot under the higher output frequency, and it has been considered as an appropriate compensation to a single camera in such navigation systems. In this master thesis, we aim to exploit a visual-inertial odometry, to further improve localization results by fusing visual data and IMU data. More precisely, we apply a loosely-coupled visual-inertial integration method to keep our system runs in a constant time.

We first discuss the types of visual SLAM. Generally, visual SLAM has two traditional approaches, which are filter-based methods and keyframe-based methods respectively. Filter-based methods marginalise all the previous camera poses to obtain the current pose estimation as well as optimized landmarks, whereas keyframe-based methods keep some of former poses and applies bundle adjustment to further improve the estimation results. We finally choose keyframe-based visual SLAM to be correction data in our Kalman filter framework because it is more efficient when scale of scene becomes larger, another reason is that it can handle loop closures more easily. We then discuss the way to integrate IMU data. We first learn *quaternion* and its operations. Instead of using Euler angle (Gimbal Lock) or rotational matrix (high memory cost), we formulate the system rotation parameters with quaternion. Followed by that, we explain the time-integration and differentiations related to the quaternions, and we suggest a error-state Kalman filter to propagate system states while tracking the system uncertainties. In order to fuse the visual data and IMU data, we suggest an adapted map scale method to automatically propagate the map scale factor, therefore solve the unknown map scale problem existed in mono visual SLAM. A keyframe-based bundle adjustment is then exploited for improving the tracking quality. Our experiments show that our suggested framework obtain more accurate and stable results compared to the state-of-art visual odometry, and our system can be easily extended to large scaled scene.

In addition to Kalman filter based framework, we briefly introduce a manifold

optimization approach to solve visual-inertial odometry problem. We have suggested a standard way to solve manifold optimization as well as several ideas to reduce the computational time in the end.

Thesis Supervisor: Roland Angst
Title: Dr.

Contents

Chapter 1

Introduction

In the past few years, the development of *Robotics* has surpassed people's expectation. The word *Robotics* first appeared in the science fiction "Liar!" by Issac Asimov [47], and it referred to the science and technology of robots. By definition, *Robotics* is a research branch that related to design, control and application of robots, as well as processing feedback from robots.

Modern robots have been classified into several categories (e.g., mobile robot, industrial robot, service robot, education robot, etc.) according to their usages. Among these categories, full-autonomous and semi-autonomous mobile robot attracts special attentions. Such robots have the abilities to move around, with or without human control. The aim of the research in mobile robot is to help us accomplish various hard tasks, whether these tasks are performed domestically, commercially or militarily. These tasks, such as assisting disabled people, defusing bombs, and repairing equipment in dangerous places are either risky or high expense for human beings.

For mobile robot, finding physical location of itself in the unknown environment is normally crucial, and such an ability (*Robot Navigation*) allows mobile robot to avoid risky obstacles and finally arrive the goal position. Roughly speaking, Robot Navigation is a computing system which processes the information from external sources (e.g., sensors), applies an algorithm to navigate robot, and sometimes builds a map of the external environment. In Robot Navigation, robots sense environmental information by their sensors. These sensors, either locally (e.g., camera, inertial measurement unit (IMU)), or globally (e.g., global positioning system (GPS)) detect environmental changes, and eventually transfer their data to robots. Generally, sensors equipped in mobile robot (local sensor) are designed to be light-weighted, small, and inexpensive considering movement convenience and low expense. Among various types of local sensors, camera and IMU sensor are considered as the most common local sensors in small mobile robot system.

A camera is a optical instrument for image captures. Modern camera has several advantages for Robot Navigation. First, the core of camera chip-set is cheap and easily installed in any mobile robot platform; Second, camera often brings rich information as it simulates the functioning of human eye. By recognizing key-points [35, 25, 34] in certain images, system observes *landmarks* in environment, and those landmarks will help to localize robots by *Triangulation* [8, 19, 10].

Figure 1-1: IMU sensor with gyroscope and accelerometer measures rotational rate and acceleration of X, Y, and Z axis regarding its local frame. Note that IMU sensor normally has bias and irreducible noises, and it is necessary to apply a calibration similar to the camera calibrations. The frequency of IMU output is normally larger than 100 [Hz]. Source: [1]

An IMU sensor (Figure 1-1) usually consists of *gyroscope*, *accelerometer*, and sometimes *magnetometer*, to measure specific force, angular rate and magnetic field regarding to its local frame. IMU is one of the major component in *Inertial Navigation System*, which is firstly used in air plane, spacecraft, guided missiles, and now also in mobile robot [4, 20, 29, 12]. IMU sensor utilizes *Dead Reckoning* to track the position of devices. Dead Reckoning integrates IMU data over time by assuming the movement model of devices is fixed (i.e., acceleration and rotation rate is constant over a small period of time).

1.1 Motivation and Contribution

The main motivation of this thesis is to improve the navigation accuracy by fusing camera data and IMU sensor data.

Single sensor-based navigation system may not fully satisfy the requirements of high-quality localization. GPS-based navigation system has been used for outdoor devices for a long time. However, such a system suffered from localizing in **in-door** environment, and also the accuracy of general GPS is not sufficient, it usually has error from 3 to 5 meters [45]. However, for a mobile robot, which may move from in-door to out-door, GPS is mostly not used, and the performance of GPS is usually served as the baseline of navigation [17].

Vision-based navigation system [8, 19], or simultaneous localization and mapping (SLAM) [9, 11, 30] gives an acceptable navigation result for mobile robot. In [8], they

recognize corner features to form the landmarks, and it uses an extended Kalman filter framework to track the uncertainty and propagates the system state, however it can not handle large-scale scene since the number of visual features will directly affect the computational time. [19] utilizes key-frame based bundle adjustment to update the map, which improves both accuracy and efficiency, though it will have some issues in large-scale scene since vision-based method often leads to a trade-off between computational complexity and localization accuracy due to the rich information and low output frequency by camera. [30] is another application of keyframe-based SLAM, however it can not handle the situation with fast-moving cameras.

Single Inertial Navigation System recognizes target pose by *Dead Reckoning* [27, 22]. The problem of Dead Reckoning is error accumulation; only few directions are observable [17] by IMU whole navigation process. When the object corrupt movement assumption, a correction step is often required.

Fusing camera data and IMU data has many advantages. On the one hand, high-frequency IMU data is an adequate compensation to low-frequency camera data; on the other hand, it is convenient to correct IMU integration by vision-based navigation results. Fusing camera data and IMU data for robot navigation is not novel. [29] applies multi-state Kalman filter (MSCKF) to update system states, which decreases the computational complexity by only storing last few keyframes. [17] discusses the inconsistency of MSCKF, and it provides a more accurate approach to propagate system states, therefore it increases the accuracy and robustness. [12] exploits a tightly-coupled way to optimize VIO problems on manifold by including image features and IMU measurements in cost function, and it increases the efficiency by eliminating support variables [7] in their factor graph.

The contributions of this thesis are as follows,

- We exploit a highly flexible, realistic tool to generate synthetic IMU sensor data and corresponding image sequences, which have provided experimental data in this thesis.
- We present the error-state Kalman filter system kinematic based on quaternion, we also explore multiple ways to integrate IMU data regarding to the different movement models.
- We propose a novel approach, key-framed bundle adjustment, to augment fusing results
- We propose a real-time visual-inertial framework implemented in C++, we demonstrate several experiments to show it is stable, scalable, high-accuracy and low-latency.

1.2 Outline of the Thesis

In Chapter 2 we first introduce the world representations and useful notations in our visual-inertial odometry. We also compare the filter method and keyframe-based method in visual SLAM problem, and in the end we explain why we choose keyframe-based method in our loosely-coupled IMU integration framework.

Chapter 3 discusses *Quaternion Algebra*. In this chapter, we introduce the basic operations on quaternion, the relationship among quaternions, rotation matrices and rotation vectors. Followed by that, we explain how to integrate or differentiate quaternion over time, which are the major tools to propagate quaternions.

Chapter 4 is the main part of this thesis. In Section 4.1, we study the error-state Kalman filter (ESKF), which is applied to IMU integration. Section 4.2 summarizes the approaches to fuse visual SLAM results into ESKF, including the adapted map scale method and keyframe-based bundle adjustment. In Section 4.3, we overview the pipeline of our visual-inertial odometry system.

Experiment results are shown in Chapter 5. First, the process of generating synthetic IMU and camera data is presented. Then we demonstrate several experiments to show that our proposed real-time visual-inertial odometry has higher accuracy and stability than the single IMU integration or visual SLAM when dealing with same data sequences. We evaluate the break-up timings of our system at last.

We briefly introduce our current researches in Chapter 6. We suggest a way to form a cost function in order to solve our visual-inertial odometry via nonlinear optimization in Section 6.1, then a standard way of manifold optimization is presented in Section 6.2.

At last, we summarize and discuss our current works and potential future works in Chapter 7.

Chapter 2

Overview of Visual-inertial Odometry

In this chapter, we will overview our visual-inertial odometry system. In section 2.1, we first introduce the world representations (e.g., world frame, camera frame and IMU frame) together with basic notations in our odometry system. In section 2.2, we then discuss two important schemes, filter method and keyframe Bundle Adjustment(keyframe BA) in SLAM algorithm, and explain why we finally choose the keyframe-based method.

2.1 World Representations and Notations

Visual-inertial odometry (VIO) [23], literally, is an odometry system, which receives environment information by visual (camera) and inertial (IMU) sensors. VIO is similar with the well-known visual odometry (VO) problem [32], with an additional IMU sensor. VIO tries to estimate agent's pose while the agent moves in the environment. One major difference between odometry and SLAM algorithm is that odometry system usually does not build a map [23], whereas SLAM algorithm simultaneously localizes and constructs a map.

To setup a VIO system, we need to first define the world representations. Globally, we have a world frame $\mathcal{W}$; world frame $\mathcal{W}$ is set to a right-handed Cartesian coordinate system that every objects has an absolute pose (translation and rotation). Each sensor has its own local frame, i.e., IMU frame $\mathcal{I}$ and camera frame $\mathcal{C}$, which is also defined as a right-handed Cartesian coordinate system. Every time sensors obtain observations within their own local frame, we estimate the pose of those sensors in world frame $\mathcal{W}$ by integrating these measurements over time. Figure 2-1 shows the overall world representations, and connections between different frames.

In this master thesis, we mainly use following notations,

- We denote scalars as a, b, c, vectors as $\mathbf{a}, \mathbf{b}, \mathbf{c}$, matrices as $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and frames as $\mathcal{A}, \mathcal{B}, \mathcal{C}$.
- We denote a measurement $\mathbf{m}$ in a particular frame $\mathcal{F}$ as $\mathbf{m}_{\mathcal{F}}$. To further simplify, any parameter that is **not** in the world frame will be denoted particularly. For example, the translation $\mathbf{p}$ in the camera frame will be denoted as $\mathbf{p}_{\mathcal{C}}$, and the translation $\mathbf{p}$ in the world frame will be simply denoted as $\mathbf{p}$.

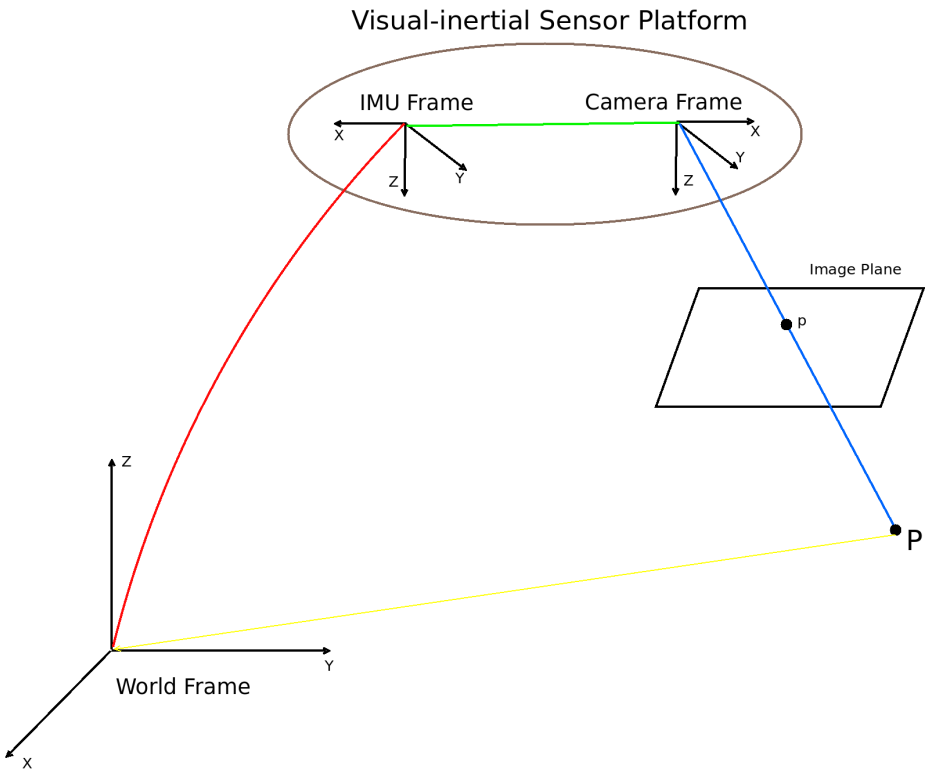
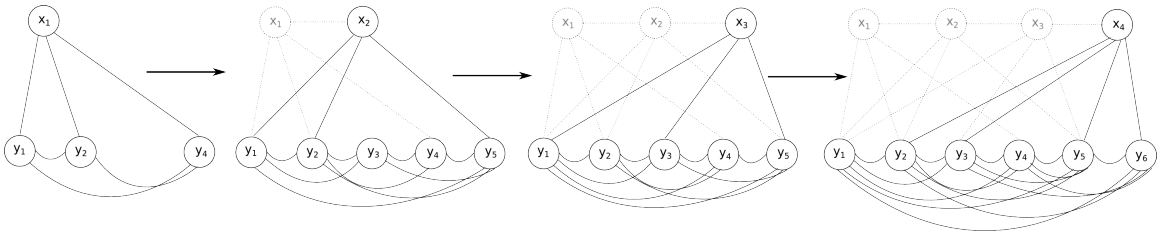
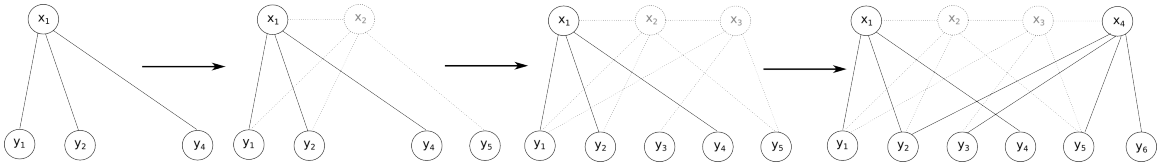


Figure 2-1: This figure shows the connection among the world frame $\mathcal{W}$, the IMU frame $\mathcal{I}$ and the camera frame $\mathcal{C}$. Green line shows the transformation between the camera and IMU, which can be pre-calibrated. Red line is the pose of IMU in the world frame. The camera frame observes point $\mathbf{p}$ of object P in the image plane, and it connects a object P by blue line. The coordinate of object P in the world frame is presented as yellow line.

- A general translation $\mathbf{t}$ should express a translation from point A to point B in frame $\mathcal{C}$, which is hereby denoted as $\mathbf{t}_{\mathcal{C}}^{AB}$. In order to simplify our notations, we denote a point $\mathbf{p}$ in frame $\mathcal{A}$ as $\mathbf{p}_{\mathcal{A}}$ if this point is the translation $\mathbf{t}_{\mathcal{A}}^{OP}$, where O is origin of frame $\mathcal{A}$, and $\mathbf{p} = P$. This holds same for vectors, which means we can directly use a point $\mathbf{p}$ to represent a vector from origin to $\mathbf{p}$.
- A general rotation is either expressed in a quaternion $\mathbf{q}$ or a rotation matrix $\mathbf{R}$. To clarify our notations, here we use quaternion $\mathbf{q}$ as an example. A quaternion is an orientation operation from frame $\mathcal{B}$ to frame $\mathcal{A}$, and it is denoted as $\mathbf{q}_{\mathcal{AB}}$ in this thesis. Note that if such a operation is from world frame $\mathcal{W}$ to a certain frame $\mathcal{B}$, we omit both frames for simplification, i.e., $\mathbf{q}_{\mathcal{BW}} \triangleq \mathbf{q}$. These notations hold same for rotation matrices.
- We use hat operator to represent the estimation of state, i.e., $\hat{\boldsymbol{x}}$ is the estimation of state $\boldsymbol{x}$.



(a) Filter



(b) Keyframe-based BA

Figure 2-2: (a) Filter method for SLAM. (b) Keyframe-based Bundle Adjustment (BA) for SLAM. We denote the i^{th} camera position as $\boldsymbol{x}_i$, i^{th} image feature as $\boldsymbol{y}_i$. We connect the line between camera and image feature if this feature is observed by this camera, the vanished observations is presented as dotted line, and the vanished camera is expressed as grey font. Both graph changes as time goes on from left to right. From (a) one can see that though only the latest camera pose is reserved, the number of edges between features are increasing fast. (b) stores some previous camera poses (keyframe) (i.e., $vecx_1$ and $\boldsymbol{x}_4$), thereby keeping the graph sparse.

2.2 Filter Versus Keyframe

It is important to note that though this thesis focuses on visual-inertial odometry for small workspaces, we intend to keep the possibility to extend our system to a general, scalable, and efficient SLAM system as well. Therefore we briefly introduce the basic concepts of SLAM here. SLAM system usually has two parallel processes, one is for localization and another is for mapping, the crucial point of building such a system is to keep both processes efficient. There are two general frameworks in visual SLAM: filter-based method and keyframe-based method. In this section, we will discuss whether filter-based method or keyframe-based method is more suitable for our case.

Filter-based SLAM [9, 10, 8] uses *extended Kalman filter* (EKF) to propagate states and update the state covariance matrices. In each step, the system obtains the current estimation of camera pose and landmark positions (3D points) by marginalising previous camera poses. This marginalization step usually eliminates the former pose and adds a few more connections to image features. As shown in Figure 2-2a, the size of graph will not grow very fast as time flows, since the former pose has been eliminated and features in environment is limited. However, once the system moves to large scale scene, it will consume more time to optimize the poses and landmarks as the graph tends to be fully-connected.

Keyframe-based SLAM [19, 29, 13, 11, 30] applies *bundle adjustment* (BA) on keyframes to update the map. In keyframe-based SLAM, it stores some historical

poses (keyframes) and image feature points to proceed a BA step. The chosen of keyframes varies from different implementations, and the idea is to choose a frame that is not very close to the last keyframe to avoid redundant information. From Figure 2-2b, it is clear that the graph stays sparse as the number of poses and features increases. Another benefit for the keyframe-based approaches is that it is more convenient to handle loop closures since the historical camera poses have been preserved.

In [37], they have shown that the computational cost for keyframe BA is $O(m^2 \cdot n)$, and for filter method is $O(n^3)$, where m is the number of key frames, and n is the number of landmarks. They conclude that keyframe-based SLAM is slightly better than filter-based SLAM with their experiment settings, especially when scale of scene becomes larger so that the number of landmarks is far more larger than the number of keyframes.

In this master thesis, we choose keyframe-based method for the visual part and filter method for the IMU integration part. Since we do not keep former information (e.g., image features or landmarks) in integration step, filter method is more efficient whereas each filter step can be regarded as an single optimization step. The reason that we use keyframe-based method for visual part is that we intend to improve the scalability of our system. Moreover the results from IMU integration are a good compensation to visual SLAM since the output frequency of IMU sensors is usually higher than visual sensors.

Chapter 3

Background on Quaternion Algebra

One important task for this master thesis is to integrate IMU data over the time in order to estimate camera poses (e.g., translations and orientations). By assuming a fixed movement model, it is straightforward to integrate translations in Cartesian space, however since rotational parameter (e.g., Euler angle, rotation matrix, or quaternion) is often represented in manifold, it is often non-trivial to integrate orientations in manifold space. In this chapter, we introduce the quaternion and the quaternion algebra, and then explore the way to operate quaternion over time. General approaches to integrating quaternion over time is given at last.

3.1 Definition of Quaternion

A quaternion Q is defined as

$$Q = q_w + q_x i + q_y j + q_z k, \tag{3.1}$$

where $\{q_w, q_x, q_y, q_z\} \in \mathbb{R}$, and $\{i, j, k\}$ are three imaginary unit lengths, i.e., $i^2 = j^2 = k^2 = ijk = -1$.

In most cases, we represent a quaternion Q as a four-element vector $\mathbf{q}$ with respect to above four real numbers $\{q_w, q_x, q_y, q_z\}$, i.e.,

$$\mathbf{q} \triangleq [q_w \ \mathbf{q}_v]^T = [q_w \ q_x \ q_y \ q_z]^T, \tag{3.2}$$

where q_w is the real part of $\mathbf{q}$, and $vecq_v$ is a 3-vector to represent imaginary part of $\mathbf{q}$.

Note that there are two different conventions of quaternion $\mathbf{q}$, *Hamilton* way [16] and *JPL* way [5]. In Hamilton convention, the real part q_w is the first component of $\mathbf{q}$, i.e., $[q_w \ \mathbf{q_v}]$, whereas in *JPL way*, the real part is the fourth component, i.e., $[\mathbf{q_v} \ q_w]$. To avoid confusions, and considering Hamilton way is more common to use, especially for implementation [14, 18], we use **Hamilton way** to represent quaternions throughout the rest of this thesis.

3.2 Properties of Quaternion

In this section, we introduce the properties of quaternion.

Summation We start with the summation of two quaternions $\mathbf{q}$ and $\mathbf{p}$

$$\boldsymbol{q} + \boldsymbol{p} = [q_w + p_w \ \mathbf{q}_v + \mathbf{p}_v]^T = [q_w + p_w \ q_x + p_x \ q_y + p_y \ q_z + p_z \ ]^T . \tag{3.3}$$

Product We use operator $\otimes$ to denote the product operation on quaternions, which gives

$$\boldsymbol{q} \otimes \boldsymbol{p} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_q + p_y q_w + p_z q_x \\ p_w q_z - p_x q_y - p_y q_z + p_z q_w \end{bmatrix} . \tag{3.4}$$

The product of two quaternions can be expressed as two equivalent matrix products

$$\boldsymbol{q_1} \otimes \boldsymbol{q_2} = Q_1^+ \boldsymbol{q_2} \tag{3.5}$$

with

$$Q_1^+ = q_w \mathbf{1} + \begin{bmatrix} 0 & -\boldsymbol{q_v}^T \\ \boldsymbol{q_v} & [\boldsymbol{q_v}]_\times \end{bmatrix}, \tag{3.6}$$

where $[\boldsymbol{q}_v]_\times$ represents the cross-product matrix of $\boldsymbol{q}_v$. The cross-product matrix of a 3-vector $\boldsymbol{v}$ is defined by

$$[\boldsymbol{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} . \tag{3.7}$$

Note that the quaternion product is not commutative, i.e.,

$$\boldsymbol{p} \otimes \boldsymbol{q} \neq \boldsymbol{q} \otimes \boldsymbol{p}. \tag{3.8}$$

However it is associative, and distributive over sum, i.e.,

$$\boldsymbol{p} \otimes (\boldsymbol{q} \otimes \boldsymbol{k}) = (\boldsymbol{p} \otimes \boldsymbol{q}) \otimes \boldsymbol{k} \tag{3.9}$$

$$\boldsymbol{p} \otimes (\boldsymbol{q} + \boldsymbol{k}) = \boldsymbol{p} \otimes \boldsymbol{q} + \boldsymbol{p} \otimes \boldsymbol{k} \tag{3.10}$$

$$(\boldsymbol{q} + \boldsymbol{k}) \otimes \boldsymbol{p} = \boldsymbol{q} \otimes \boldsymbol{p} + \boldsymbol{k} \otimes \boldsymbol{p}. \tag{3.11}$$

Conjugate The conjugate of a quaternion $\boldsymbol{q}^*$ is defined by

$$\boldsymbol{q}^* \triangleq q_w - \boldsymbol{q_v} = [q_w \ \ -\boldsymbol{q_v}]^T . \tag{3.12}$$

Identity An identity quaternion $\mathbf{q}$ is defined as, for any given quaternion $\mathbf{p}$,

$\boldsymbol{q} \otimes \boldsymbol{p} = \boldsymbol{p} \otimes \boldsymbol{q} = \boldsymbol{p}$. An identical quaternion $\mathbf{q}$ satisfies that

$$\boldsymbol{q} = [1\ 0\ 0\ 0]^T. \tag{3.13}$$

In this master thesis, we denote identity quaternion as $\boldsymbol{q_1}$.

Norm The norm of a quaternion $\|\boldsymbol{q}\|$ is defined similar to the norm of a vector, which is

$$\|\boldsymbol{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}. \tag{3.14}$$

Inverse The inverse of a quaternion $\boldsymbol{q}^{-1}$ is defined as

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^* / \|\boldsymbol{q}\|, \tag{3.15}$$

which leads to

$$\boldsymbol{q}^* \otimes \boldsymbol{q}^{-1} = \boldsymbol{q}^{-1} \otimes \boldsymbol{q}^* = \boldsymbol{q}_{\mathbb{I}}. \tag{3.16}$$

Unit quaternion The norm of a unit quaternion $\|\boldsymbol{q}\|$ is 1, and the inverse of such a unit quaternion is equal to the conjugate of this quaternion

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^*. \tag{3.17}$$

Pure quaternion A pure quaternion $\boldsymbol{q}$ is defined as

$$\boldsymbol{q} = [0\ \boldsymbol{q}_v]^T = [0\ q_x\ q_y\ q_z]^T. \tag{3.18}$$

Let pure quaternion $\boldsymbol{q} = \theta \boldsymbol{u}$, where $\theta = \|\boldsymbol{q}\|$, we can compute the exponential of $\boldsymbol{q}$ with the help of Euler formula

$$\mathrm{e}^{\boldsymbol{q}} = \mathrm{e}^{\theta \boldsymbol{u}} = \cos\theta + \boldsymbol{u}\sin\theta = [\cos\theta\ \boldsymbol{u}\sin\theta]^T, \tag{3.19}$$

which is still a quaternion, and moreover $\mathrm{e}^{\boldsymbol{q}}$ is a unit quaternion because its norm satisfies $\|\mathrm{e}^{\boldsymbol{q}}\|^2 = \cos\theta^2 + \sin\theta^2 = 1$.

3.3 Quaternions and Rotation operations

We discuss the relationship between quaternions and rotation operations by first introducing rotation vector $\boldsymbol{v}$.

Given a rotation vector $\boldsymbol{v} = \phi \boldsymbol{u}$, where ϕ is the norm of $\boldsymbol{v}$ and $\boldsymbol{u}$ is a unit vector, we can rotate a vector $\boldsymbol{x}$ by an angle ϕ around the axis $\boldsymbol{u}$ following right-handed rule, and obtain a new vector $\boldsymbol{x'}$

$$\boldsymbol{x'} = \boldsymbol{x}_{||} + \boldsymbol{x}_{\perp}\cos\phi + (\boldsymbol{u} \times \boldsymbol{x})\sin\phi, \tag{3.20}$$

where $\boldsymbol{x}_{||} = (\boldsymbol{x} \cdot \boldsymbol{u})\boldsymbol{u}$ is the component parallel to $\boldsymbol{x}$, and $\boldsymbol{x}_{\perp} = -\boldsymbol{u} \times (\boldsymbol{u} \times \boldsymbol{x})$ is the component perpendicular to $\boldsymbol{x}$, therefore $\boldsymbol{x} = \boldsymbol{x}_{||} + \boldsymbol{x}_{\perp}$. This formula is known as *vector rotation formular* or *Rodrigues formula*.

Then we can define a rotation matrix $\mathbf{R}$ by rotation vector $\boldsymbol{v} = \phi \boldsymbol{u}$ with the help

of Equation (3.7) as

$$\mathbf{R} = \mathrm{e}^{[\boldsymbol{v}]_\times}. \tag{3.21}$$

We can rotate a vector $\boldsymbol{x}$ by an angle ϕ around the axis $\boldsymbol{u}$ using $\mathbf{R}$ in a clean way

$$\boldsymbol{x'} = \mathbf{R}\boldsymbol{x}, \tag{3.22}$$

furthermore one can show that result in Equation (3.22) is equivalent with the result in Equation (3.20) [43].

Constructing a unit quaternion $\boldsymbol{q}$ by Equation (3.19) with a rotation vector $\boldsymbol{v} = \phi\boldsymbol{u}$ as

$$\boldsymbol{q} = \mathrm{e}^{\boldsymbol{v}/2} = \begin{bmatrix} \cos\phi/2 \\ \boldsymbol{u}\sin\phi/2 \end{bmatrix}, \tag{3.23}$$

we can rotate a vector $\boldsymbol{x}$ by an angle ϕ around the axis $\boldsymbol{u}$

$$\boldsymbol{x'} = \boldsymbol{q} \otimes \boldsymbol{x} \otimes \boldsymbol{q}^*. \tag{3.24}$$

We then show $\boldsymbol{x'}$ in Equation (3.24) is equivalent with $\boldsymbol{x'}$ in Equation (3.20). First we transferred the vector $\boldsymbol{x}$ into pure quaternion form as

$$\boldsymbol{x}_q = [0\ \boldsymbol{x}]^T, \tag{3.25}$$

then we rewrite formula (3.24) as

$$\begin{bmatrix} 0 \\ \boldsymbol{x'} \end{bmatrix} = \begin{bmatrix} \cos\phi/2 \\ \boldsymbol{v}\sin\phi/2 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \boldsymbol{x} \end{bmatrix} \otimes \begin{bmatrix} \cos\phi/2 \\ -\boldsymbol{v}\sin\phi/2 \end{bmatrix}. \tag{3.26}$$

Expanding it by Equation (3.4), it is easily to show that

$$\boldsymbol{x'} = \boldsymbol{x}_{||} + \boldsymbol{x}_{\perp}\cos\phi + (\boldsymbol{u} \times \boldsymbol{x})\sin\phi, \tag{3.27}$$

which is exactly Equation (3.20).

To summarize here, we can construct a quaternion $\boldsymbol{q}$ or a rotation matrix $\mathbf{R}$ by any rotation vector $\boldsymbol{v} = \phi\boldsymbol{u}$, we denote such a quaternion as $\boldsymbol{q}\{\boldsymbol{v}\}$ and rotation matrix as $\mathbf{R}\{\boldsymbol{v}\}$ respectively. A rotation operation of a vector $\boldsymbol{x}$ related to $\boldsymbol{v}$ can either be expressed as a quaternion $\boldsymbol{q}\{\boldsymbol{v}\} \otimes \boldsymbol{x} \otimes \boldsymbol{q}\{\boldsymbol{v}\}^*$, or as a rotation matrix $\mathbf{R}\{\boldsymbol{v}\}\boldsymbol{x}$. Note that we simplify $\mathbf{R}\{\boldsymbol{v}\}$ to $\mathbf{R}$, and/or $\boldsymbol{q}\{\boldsymbol{v}\}$ to $\boldsymbol{q}$ in the rest of this master thesis.

It is also easy to show the conversion from a rotation matrix $\mathbf{R}$ to a quaternion $\boldsymbol{q}$, which we use in this thesis. Knowing that

$$\boldsymbol{q} \otimes \boldsymbol{x} \otimes \boldsymbol{q}^* = \mathbf{R}\boldsymbol{x}, \tag{3.28}$$

we can construct $\mathbf{R} = \mathbf{R}\{\boldsymbol{q}\}$ by

$$\mathbf{R} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_xq_y - q_wq_z) & 2(q_xq_z + q_wq_y) \\ 2(q_xq_y + q_wq_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_yq_z - q_wq_x) \\ 2(q_xq_z - q_wq_y) & 2(q_yq_z + q_wq_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}. \tag{3.29}$$

The conversion from rotation matrix to quaternion, which is beyond the content of this thesis, can be found in [41].

3.4 Quaternion Differentiation

We introduce the differentiation on quaternion by first define

$$\boldsymbol{q}(t+\Delta t) \triangleq \boldsymbol{q}(t) \otimes \Delta \boldsymbol{q}, \tag{3.30}$$

where $\boldsymbol{q}(t)$ is the quaternion at time t and $\Delta \boldsymbol{q}$ is quaternion transformation within a small period time Δt.

One can expand $\Delta \boldsymbol{q}$ by Taylor expansions with Equation (3.23) to

$$\Delta \boldsymbol{q} = \begin{bmatrix} 1 \\ \frac{1}{2}\Delta \boldsymbol{\theta} \end{bmatrix} + O(\|\Delta \boldsymbol{\theta}\|^2), \tag{3.31}$$

where $\Delta \boldsymbol{\theta}$ is a angular vector corresponding to $\Delta \boldsymbol{q}$. In fact, the angular rate $\boldsymbol{\omega}$ at time t is defined as

$$\boldsymbol{\omega}(t) \triangleq \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{\theta}}{\Delta t}, \tag{3.32}$$

which is one of measurements we can obtain from IMU sensor.

By definition of the derivative, we can obtain the time-derivative $\dot{\boldsymbol{q}}$ of quaternion $\boldsymbol{q}$ as

$$\dot{\boldsymbol{q}} = \frac{d\boldsymbol{q}(t)}{dt} \triangleq \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t)}{\Delta t}, \tag{3.33}$$

which follows

$$\begin{aligned} \dot{\boldsymbol{q}} &\triangleq \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\boldsymbol{q} \otimes \Delta \boldsymbol{q} - \boldsymbol{q}}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\boldsymbol{q} \otimes (\begin{bmatrix} 1 \\ \frac{1}{2}\Delta \boldsymbol{\theta} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix})}{\Delta t} \\ &= \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \end{aligned} \tag{3.34}$$

Here we simplify $\boldsymbol{q}(t)$ to $\boldsymbol{q}$. Then we can obtain the time-derivative on quaternion by writing angular rate into pure quaternion form (3.18), which is

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}. \tag{3.35}$$

3.5 Time-integration on Quaternion

To integrate quaternion over time, we explore the relationship between $\boldsymbol{q}(t_n)$ and $\boldsymbol{q}(t_{n+1})$ where $t_n = n\Delta t$. Expanding $\boldsymbol{q}(t_{n+1})$ using Taylor series, we have

$$\boldsymbol{q}(t_{n+1}) = \boldsymbol{q}(t_n) + \dot{\boldsymbol{q}}(t_n)\Delta t + \frac{1}{2!}\ddot{\boldsymbol{q}}(t_n)\Delta t^2 + \frac{1}{3!}\dddot{\boldsymbol{q}}(t_n)\Delta t^3 + \dots. \tag{3.36}$$

Assume that the second order derivative of rotational rate is zero, which is $\ddot{\boldsymbol{\omega}} = 0$, we have

$$\dot{\boldsymbol{q}}(t_{n+1}) = \frac{1}{2}\boldsymbol{q}(t_n) \otimes \boldsymbol{\omega}(t_n) \tag{3.37}$$

$$\ddot{\boldsymbol{q}}(t_{n+1}) = \frac{1}{2^2}\boldsymbol{q}(t_n) \otimes \boldsymbol{\omega}^2(t_n) + \frac{1}{2}\boldsymbol{q}(t_n) \otimes \dot{\boldsymbol{\omega}} \tag{3.38}$$

$$\dddot{\boldsymbol{q}}(t_{n+1}) = \frac{1}{2^3}\boldsymbol{q}(t_n) \otimes \boldsymbol{\omega}^3(t_n) + \frac{1}{4}\boldsymbol{q}(t_n) \otimes \dot{\boldsymbol{\omega}}\boldsymbol{\omega}(t_n) + \frac{1}{2}\boldsymbol{q}(t_n) \otimes \boldsymbol{\omega}(t_n)\dot{\boldsymbol{\omega}} \tag{3.39}$$

$$\vdots$$

and so forth and so on. We then get the result of time integration by taking Equation (3.37), (3.38), and (3.39) back into Equation (3.36).

We hereby uses a stronger assumption that angular rate $\boldsymbol{\omega}(t_n)$ remains constant during a small time period Δt, which is $\dot{\boldsymbol{\omega}} = 0$. Considering the sampling rate of IMU sensor is usually high (> 100 [Hz]), this assumption is actually quite accurate in practice. Moreover it gives us a cleaner expression of the quaternion time integration formula.

Given $\dot{\boldsymbol{\omega}} = 0$, we can get

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes (1 + \frac{1}{2}\boldsymbol{\omega}_n\Delta t + \frac{1}{2!}(\frac{1}{2}\boldsymbol{\omega}_n\Delta t)^2 + \frac{1}{3!}(\frac{1}{2}\boldsymbol{\omega}_n\Delta t)^3 + \frac{1}{4!}(\frac{1}{2}\boldsymbol{\omega}_n\Delta t)^4 + \dots). \tag{3.40}$$

Here we regard $\boldsymbol{q}$ and $\boldsymbol{\omega}$ as series, which is exactly

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes \mathrm{e}^{\boldsymbol{\omega}\Delta t/2}. \tag{3.41}$$

We can rewrite it by Equation (3.23) as

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes \boldsymbol{q}\{\boldsymbol{\omega}_n\Delta t\}, \tag{3.42}$$

which is called ***Zeroth order forward integration*** of quaternion over time.

We can obtain ***Zeroth order backward integration*** by assuming the angular rate remains $\boldsymbol{\omega}_{n+1}$ within Δt, then we have

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes \boldsymbol{q}\{\boldsymbol{\omega}_{n+1}\Delta t\}, \tag{3.43}$$

and ***Zeroth order midward integration*** by assuming the angular rate holds $\bar{\boldsymbol{\omega}}_{n+1} =$

$(\boldsymbol{\omega}_n + \boldsymbol{\omega}_{n+1})/2$ within Δt, which is

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes \boldsymbol{q}\{\bar{\boldsymbol{\omega}}_n \Delta t\}. \tag{3.44}$$

Though not used in this master thesis, we notice that [39] gives ***First order integration*** by assuming angular rate is linear with time, i.e., $\dot{\boldsymbol{\omega}} = \frac{\boldsymbol{\omega}_{n+1} - \boldsymbol{\omega}_n}{\Delta t}$, which is precisely

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n \otimes \boldsymbol{q}\{\bar{\boldsymbol{\omega}}_n \Delta t\} + \frac{\Delta t^2}{48} \boldsymbol{q}_n \otimes (\boldsymbol{\omega}_n \otimes \boldsymbol{\omega}_{n+1} - \boldsymbol{\omega}_{n+1} \otimes \boldsymbol{\omega}_n) + \dots \tag{3.45}$$

in our quaternion convention.

Chapter 4

Modular Sensor Fusing

In the previous chapters, we learned how to represent each frame and analysed the difference between filter method and keyframe BA method for our odometry system; we also learned quaternion algebra and basic approaches for quaternion integration and derivative over the time under some general assumptions. In this chapter, we will explore the details of our sensor fusing approach, which uses so called *IMU loosely-coupled integration framework* [42]. In such a framework, system propagates states via Kalman filter (KF) based on IMU measurements, and extrasensory (e.g., camera, GPS etc.) data are used in correction step. The computational cost for Kalman-filter-styled approach is usually linear, hence IMU loosely-coupled integration framework provides a better trade-off between computational complexity and accuracy in a real-time robotic navigation system.

4.1 Error-state Kalman Filter for IMU Integration

The error-state Kalman filter (ESKF) follows the paradigm of Kalman filter, which also has prediction step and correction step. However, ESKF separates system into three different states: true state, nominal state, and error state. Nominal state processes large signal, which is integrable in a non-linear fashion, whereas error state keeps track of error and noise term, which can be integrated in a linear way. The composition of nominal state and error state is called true state, which is the realistic status of the system.

The ESKF has several beneficial properties when building a visual-inertial odometry:

1. The computation of Jacobian is often very fast, because the error state is small and all second order products are negligible. This is an important factor for building a real-time system.

2. Fusing visual data with IMU data is straightforward in Kalman filter correction step. One can utilize the tracking result to correct the IMU integration state.

3. Large signals have been integrated into nominal states, so that we can apply the correction step in a lower rate than prediction step.

The procedure of ESKF in this system can be explained as follows. Firstly, IMU data is integrated into nominal state via numerical integration methods, note that nominal state does not take noise terms or error terms into account, hence nominal state will accumulate errors continuously. The error state then predicts the errors and noise terms using general extended KF paradigm, meaning it will predict the mean and covariance of the error state. In parallel a correction step is performed at a lower rate, the results of visual tracking are used to correct the error state, the error state is then injected into nominal state, which the nominal state becomes a final estimation of system at that point. The system continues until the termination conditions have been met.

We explain the ESKF for IMU integration in this section, the derivations are partially based on the instructions in [36]. Visual sensor as the correction data is introduced in Section 4.2, which is inspired by [42].

4.1.1 System Kinematics

We denote our true state $\boldsymbol{x}_t$ as

$$\boldsymbol{x}_t = \boldsymbol{x}_n \oplus \boldsymbol{x}_e, \tag{4.1}$$

where $\boldsymbol{x}_n$ is the nominal state for large signals, $\boldsymbol{x}_e$ is error state for small error/noise signal, and we have used $\oplus$ to denote a general composition step.

We then introduce position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, quaternion $\boldsymbol{q}$, accelerometer bias $\boldsymbol{a}_b$, gyroscope bias $\boldsymbol{\omega}_b$ and gravity vector $\boldsymbol{g}$ into true state, nominal state and error state respectively. The general composition step can be shown as

$$\begin{aligned}
\boldsymbol{p}_t &= \boldsymbol{p}_n + \boldsymbol{p}_e && (4.2)\\
\boldsymbol{v}_t &= \boldsymbol{v}_n + \boldsymbol{v}_e && (4.3)\\
\boldsymbol{q}_t &\approx \boldsymbol{q}_n \otimes \begin{bmatrix} 1 \\ \boldsymbol{\theta}_e/2 \end{bmatrix} && (4.4)\\
\boldsymbol{a}_{bt} &= \boldsymbol{a}_{bn} + \boldsymbol{a}_{be} && (4.5)\\
\boldsymbol{\omega}_{bt} &= \boldsymbol{\omega}_{bn} + \boldsymbol{\omega}_{be} && (4.6)\\
\boldsymbol{g}_t &= \boldsymbol{g}_n + \boldsymbol{g}_e, && (4.7)
\end{aligned}$$

note that we derive Equation (4.4) by the small angle approximation (Equation (3.31)). We apply angular error $\boldsymbol{\theta}_e$ instead of quaternion error in error state following classical approaches.

We then construct kinematic equations for true state, which are

$$\dot{\boldsymbol{p}}_t = \boldsymbol{v}_t \tag{4.8}$$
$$\dot{\boldsymbol{v}}_t = \mathbf{R}_t(\boldsymbol{a}_m - \boldsymbol{a}_{bt} - \boldsymbol{a}_n) + \boldsymbol{g}_t \tag{4.9}$$
$$\dot{\boldsymbol{q}}_t = \frac{1}{2}\boldsymbol{q}_t \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bt} - \boldsymbol{\omega}_n) \tag{4.10}$$
$$\dot{\boldsymbol{a}_{bt}} = \boldsymbol{a}_w \tag{4.11}$$
$$\dot{\boldsymbol{\omega}_{bt}} = \boldsymbol{\omega}_w \tag{4.12}$$
$$\dot{\boldsymbol{g}}_t = 0, \tag{4.13}$$

where $\boldsymbol{a}_m$ and $\boldsymbol{\omega}_m$ are the measurements from accelerometer and gyroscope respectively within **local frame**, $\boldsymbol{a}_n$ and $\boldsymbol{\omega}_n$ are noises with those measurements, $\boldsymbol{a}_w$ and $\boldsymbol{\omega}_w$ are white Gaussian noise together with accelerometer and gyroscope bias, and $\mathbf{R}_t$ is the rotation matrix corresponding to true state quaternion, i.e., $\mathbf{R}_t \triangleq \mathbf{R}_t\{\boldsymbol{q}\}$ regarding to Equation (3.29). We use similar notations in nominal state and error state.

We obtain kinematic equations for nominal state by ignoring all small signals, which leads to

$$\dot{\boldsymbol{p}}_n = \boldsymbol{v}_n \tag{4.14}$$
$$\dot{\boldsymbol{v}}_n = \mathbf{R}_n(\boldsymbol{a}_m - \boldsymbol{a}_{bt}) + \boldsymbol{g}_n \tag{4.15}$$
$$\dot{\boldsymbol{q}}_n = \frac{1}{2}\boldsymbol{q}_n \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn}) \tag{4.16}$$
$$\dot{\boldsymbol{a}_{bn}} = 0 \tag{4.17}$$
$$\dot{\boldsymbol{\omega}_{bn}} = 0 \tag{4.18}$$
$$\dot{\boldsymbol{g}}_n = 0, \tag{4.19}$$

and the error state with small signals is

$$\dot{\boldsymbol{p}}_e = \boldsymbol{v}_e \tag{4.20}$$
$$\dot{\boldsymbol{v}}_e = \mathbf{R}_n \left[\boldsymbol{a}_m - \boldsymbol{a}_{bn}\right]_\times - \mathbf{R}_n \boldsymbol{a}_{be} + \boldsymbol{g}_e - \mathbf{R}_n \boldsymbol{a}_n \tag{4.21}$$
$$\dot{\boldsymbol{\theta}}_e = \left[\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn}\right]_\times - \boldsymbol{\omega}_{be} - \boldsymbol{\omega}_n \tag{4.22}$$
$$\dot{\boldsymbol{a}_{be}} = \boldsymbol{a}_w \tag{4.23}$$
$$\dot{\boldsymbol{\omega}_{be}} = \boldsymbol{\omega}_w \tag{4.24}$$
$$\dot{\boldsymbol{g}}_e = 0. \tag{4.25}$$

It is trivial to derive Equation (4.20, 4.23, 4.24, 4.25), see Appendix A for derivation of Equation (4.21 and 4.22).

4.1.2 State Time-integration and Error-state Jacobian

We propose time-integration equations between any two time stamp t_n and t_{n+1} where we measure the time difference Δt as $\Delta t = t_{n+1} - t_n$. In order to simplify our notations,

we denote the last state parameters as $\boldsymbol{x}$, and denote the current state parameters as $\boldsymbol{x}'$, where the current state is measured at time stamp t_n, and last state is measured at t_{n-1}. Same notations are set for error state. Therefore, time-integration equations for nominal state for one update are

$$\begin{aligned}
\boldsymbol{p}'_n =& \boldsymbol{p}_n + \boldsymbol{v}_n \Delta t + \frac{1}{2}(\mathbf{R}_n(\boldsymbol{a}_m - \boldsymbol{a}_{bt}) + \boldsymbol{g}_n)\Delta t^2 && (4.26)\\
\boldsymbol{v}'_n =& \boldsymbol{v}_n + (\mathbf{R}_n(\boldsymbol{a}_m - \boldsymbol{a}_{bt}) + \boldsymbol{g}_n)\Delta t && (4.27)\\
\boldsymbol{q}'_n =& \boldsymbol{q}_n \otimes \boldsymbol{q}\{(\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn})\Delta t\} && (4.28)\\
\boldsymbol{a}'_{bn} =& \boldsymbol{a}_{bn} && (4.29)\\
\boldsymbol{\omega}'_{bn} =& \boldsymbol{\omega}_{bn} && (4.30)\\
\boldsymbol{g}'_n =& \boldsymbol{g}_n. && (4.31)
\end{aligned}$$

We use ***Zeroth order forward integration*** explained in Section 3.5 to integrate our state over time, this is also called (explicit) Euler method in Runge-Kutta numerical integration methods (see Appendix B.1).

We integrate our error state in the same manner, except we have truncated second-order signal out. We obtain the integration equations for error state by

$$\begin{aligned}
\boldsymbol{p}'_e =& \boldsymbol{p}_e + \boldsymbol{v}_e \Delta t && (4.32)\\
\boldsymbol{v}'_e =& \boldsymbol{v}_e + (-\mathbf{R}_n \left[\boldsymbol{a}_m - \boldsymbol{a}_{bn}\right]_\times \boldsymbol{\theta}_e - \mathbf{R}_n \boldsymbol{a}_{be} + \boldsymbol{g}_e)\Delta t + \boldsymbol{v}_i && (4.33)\\
\boldsymbol{\theta}'_e =& (\mathbf{R}_n^T\{\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn}\}\boldsymbol{\theta}_e - \boldsymbol{\omega}_{be})\Delta t + \boldsymbol{\theta}_i && (4.34)\\
\boldsymbol{a}'_{be} =& \boldsymbol{a}_{be} + \boldsymbol{a}_i && (4.35)\\
\boldsymbol{\omega}'_{be} =& \boldsymbol{\omega}_{be} + \boldsymbol{\omega}_i && (4.36)\\
\boldsymbol{g}'_e =& \boldsymbol{g}_e, && (4.37)
\end{aligned}$$

where $\boldsymbol{v}_i$, $\boldsymbol{\theta}_i$, $\boldsymbol{a}_i$, and $\boldsymbol{\omega}_i$ are random impulses for velocity, angular error, accelerometer bias and gyroscope. Those impulses can be modelled by Gaussian process. We derive Equation (4.34) by close-formed integration methods described in Appendix B.2.

We then provide the Jacobian of error state $\mathbf{J}_{x_e}$ for ESKF prediction step usage

$$\mathbf{J}_{e'e} = \frac{\partial \boldsymbol{x}'_e}{\partial \boldsymbol{x}_e} = \begin{bmatrix}
\mathbf{1} & \mathbf{1}\Delta t & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & \mathbf{R}_n \left[\boldsymbol{a}_m - \boldsymbol{a}_{bn}\right]_\times \Delta t & \mathbf{R}_n \Delta t & 0 & \mathbf{1}\Delta t \\
0 & 0 & \mathbf{R}_n^T\{\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn}\}\Delta t & 0 & -\mathbf{1}\Delta t & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1}
\end{bmatrix}. \tag{4.38}$$

Note that partial derivative between true state $\boldsymbol{x}_t$ and $\boldsymbol{x}_e$ is not identity because we use different parameters to represent orientations, e.g., quaternions in true state and nominal state, angular errors in error state. We then give the Jacobian of true

state with respect to error state with

$$\mathbf{J}_{te} = \frac{\partial \boldsymbol{x}_t}{\partial \boldsymbol{x}_e} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial \boldsymbol{q}_t}{\partial \boldsymbol{\theta}_e} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}. \tag{4.39}$$

By Equation (3.6), we have

$$\frac{\partial \boldsymbol{q}_t}{\partial \boldsymbol{\theta}_e} = \frac{1}{2} Q^+(\boldsymbol{q}) \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.40}$$

$$= \frac{1}{2} \begin{bmatrix} -q_x & -q_y & -q_z \\ q_w & -q_x & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \end{bmatrix}. \tag{4.41}$$

4.1.3 State Propagation

Initially, nominal state $\boldsymbol{x}_n$ has been set based on the prior knowledge, and there is no error at start, i.e., all terms in the error state are set to zero. We assume error state $\boldsymbol{x}_e$ as a normal distribution, i.e., $\boldsymbol{x}_e \sim \mathcal{N}(\hat{\boldsymbol{x}}_e, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ denotes the covariance matrix for error state, which helps us to track the uncertainty of error state. Note that $\boldsymbol{\Sigma}$ is initialized to a very small diagonal matrix.

At the certain round, we first obtain the measurements from our accelerator and gyroscope, and compute a new nominal state estimation $\hat{\boldsymbol{x}_n'}$ by Equation (4.26) to (4.31). We then compute error state Jacobian $\mathbf{J}'_{e'e}$ by Equation (4.38), then update the error state and covariance matrix of current error state by

$$\hat{\boldsymbol{x}_e'} = \mathbf{J}'_{e'e} \hat{\boldsymbol{x}}_e \tag{4.42}$$

$$\boldsymbol{\Sigma}' = \mathbf{J}'_{e'e} \boldsymbol{\Sigma} (\mathbf{J}'_{e'e})^T, \tag{4.43}$$

which is called *prediction step* in ESKF. We omit prime symbol in the next step, i.e., current state $\boldsymbol{x}$ is replaced by $\boldsymbol{x}'$.

We then assume correction measurement $\boldsymbol{y}$ from extrasensory data is a non-linear function with additional white Gaussian noise $w \sim \mathcal{N}(0, \mathbf{W})$ of our true state, i.e.,

$$\boldsymbol{y} = h(\boldsymbol{x}_t) + w, \tag{4.44}$$

and the *correction step* of ESKF are as follows

$$\mathbf{K} = \mathbf{\Sigma}\mathbf{H}^T(\mathbf{H}\mathbf{\Sigma}\mathbf{H}^T + \mathbf{W})^{-1} \tag{4.45}$$

$$\hat{\boldsymbol{x}}_e' = \mathbf{K}(\boldsymbol{y} - h(\hat{\boldsymbol{x}}_t)) \tag{4.46}$$

$$\mathbf{\Sigma}' = (\mathbf{1} - \mathbf{K}\mathbf{H})\mathbf{\Sigma}, \tag{4.47}$$

where $\mathbf{H}$ is the Jacobian matrix of measurement function $h(\cdot)$ with respect to error state $\boldsymbol{x}_e$ (see Section 4.2.2). Note the estimation of true state here is the nominal state since we have not observed the mean of error state. The true state is estimated by Equation (4.1) and Equation (4.8) to (4.13). Depending on the output frequency of extra sensor, *correction step* often happens on a lower rate than *prediction step*. As always, we omit prime symbol in next few steps as state has been refreshed.

Before the system enters into the next round, we reset error state to initial state, i.e., $\hat{\boldsymbol{x}}_e = 0$ in our case. We update the covariance matrix $\mathbf{\Sigma}$ by

$$\mathbf{\Sigma}' = \mathbf{J}_{ge}\mathbf{\Sigma}(\mathbf{J}_{ge})^T, \tag{4.48}$$

where $\mathbf{J}_{ge}$ is Jacobian matrix of updated error state with respect to old error state. $\mathbf{J}_{ge}$ is given by

$$\mathbf{J}_{ge} \triangleq \frac{\partial \boldsymbol{x}_e'}{\partial \boldsymbol{x}_e} = \begin{bmatrix} \mathbf{1}_6 & 0 & 0 \\ 0 & \mathbf{1} - \left[\frac{1}{2}\hat{\boldsymbol{\theta}}_e\right]_\times & 0 \\ 0 & 0 & \mathbf{1}_9 \end{bmatrix}, \tag{4.49}$$

where we use $\boldsymbol{x}'$ to denote the new state after resetting error state. The Jacobian is identical on all diagonal blocks except angular error term. We derive the $\mathbf{J}_{ge}$ by two observations, first the true state will not be able to change during the reset of error state; second, the new nominal state is obtained by injecting the error state from last nominal state. These two observations can be transformed into following equations

$$\boldsymbol{x}_n' \oplus \boldsymbol{x}_e' = \boldsymbol{x}_n \oplus \boldsymbol{x}_e \tag{4.50}$$

$$\boldsymbol{x}_n' = \boldsymbol{x}_n \oplus \hat{\boldsymbol{x}}_e. \tag{4.51}$$

Putting Equation (4.51) into Equation (4.50), we have

$$\boldsymbol{q}_e' = (\boldsymbol{q}_n \otimes \hat{\boldsymbol{q}}_e)^* \otimes \boldsymbol{q}_n \otimes \boldsymbol{q}_e = Q^+(\hat{\boldsymbol{q}}_e)^*\boldsymbol{q}_e. \tag{4.52}$$

By using small angle approximation (Equation (3.6)) and ignoring the second-order term, we can obtain

$$\boldsymbol{\theta}_e' = -(\hat{\boldsymbol{\theta}}_e + (\mathbf{1} - \left[\frac{1}{2}\hat{\boldsymbol{\theta}}_e\right]_\times)\boldsymbol{\theta}_e. \tag{4.53}$$

After derivation of old angular term from both sides, we get

$$\frac{\partial \boldsymbol{\theta}_e'}{\partial \boldsymbol{\theta}_e} = \mathbf{1} - \left[\frac{1}{2}\hat{\boldsymbol{\theta}}_e\right]_\times, \tag{4.54}$$

which is precisely the term in Equation (4.49).

4.2 Camera as Complementary Sensory Data

As we discussed in Section 4.1, we need extrasensory data in ESKF *correction step*, and we modelled this sensor as a non-linear measurement of true state $\boldsymbol{x}_t$ plus a white Gaussian noise as shown in Equation (4.44). This sensor in our case should satisfy the following conditions:

- This sensor should carry rich information as we need to obtain accurate camera pose estimation from it.
- This sensor, unlike GPS, should work both inside and outside environment as our system is designed for mobile robots.
- This sensor should be light-weight, low-expense and better easy to handle as this is the general requirements for mobile robots.

Above conditions lead to our choice — camera. It is worthy to note that though we treat data in correction step as a black box, it is also possible to choose a multiple sensory platform. However on the one hand, we choose camera not only it satisfies all the above conditions, also the camera as the most commonly-used sensor has been well-investigated and well-understand. It is more convenient to find multiple possible approaches to solve our issues, which is precisely camera pose estimation in our case; on the other hand, multiple extra sensors (i.e., camera+GPS) probably meet synchronization issues. Thereby we choose the camera images as complementary data in this work.

4.2.1 Introduction to Monocular Visual Odometry

Visual odometry estimates the camera pose (e.g., global translation and rotation) we require in ESKF prediction step as the transformation between camera and IMU usually has been pre-calibrated. Though there are other systems such as stereo-based odometry [28], depth-camera odometry [31] in visual odometry categories, however considering we only use single camera in this work, we here mainly introduce different types of monocular visual odometry.

A monocular odometry usually works as follows. Initially, the system obtains the first estimation of camera's pose by *Homography* [46] between two images. After a new image is acquired, the system then tries to find the correspondences among common *landmarks* in different views, where a landmark is defined as the most distinctive object in the real world.

Depending on the types of correspondences, we have

- **Feature-based methods** As proposed in [9, 10, 19, 30], feature-based methods use image features to denote the correspondences among landmarks, e.g., 3D points belongs to the same if the features corresponding to it are same. The system then reprojects similar points from the last image to the current one

using estimated camera pose transformation, and errors between points in the current visual frame is called *reprojection error*.

- **Direct methods** As proposed in [11], it uses image intensity (e.g., photometric error) to find such correspondences.
- **Semi-direct methods** As shown in [13], it uses combinations of image features and image intensities to find correspondences.

After the connections of points between two images have been established, camera pose is estimated by minimizing the re-projection error and/or image intensity error.

Building a map based on such a odometry is rather straightforward. A point is recognized as a part of landmark if it has been frequently tracked, and then it will be inserted into the map as a recognized *map point*; a map is then constructed by these map points, and normally a batch processing (e.g., bundle adjustment) is used for optimizing the map during mapping process. However, such a technique can also be used to improve the pose estimation quality as we discuss in Section 4.2.3.

4.2.2 Self-adapt Map Scale

The estimation of camera pose from monocular visual odometry (VO) is usually a good compensation to ESKF IMU integration since global translation is unobservable to IMU integration. Though camera is an angle-sensor, it is impossible to obtain the real map scale [13] since the global translation from monocular VO has been scaled up/down together with real world scale. Researchers have tried to estimate map scale factor by aligning the first few frames with ground-truth data [13], or initialize map scale by a standard object (e.g., a A4 paper) [9], but it is still likely to accumulate scale drift with time goes on. In our framework, since IMU measures under the real world scale, we can propagate the map scale in our ESKF framework by introducing a scale factor in our state representations.

We first add the scale factor $\boldsymbol{\lambda}$ which contains a 3-vector to represent the scalability with x-axis, y-axis, and z-axis respectively. The advantage of 3-vector rather than a scalar is that we can somehow avoid the crazy jumps even if the differences between two successive measurements is very small in certain axis, i.e., a micro helicopter flies with a unchanged height. Now our true state, nominal state, and error state have been changed into

$$\boldsymbol{x}_t = [\boldsymbol{p}_t \ \boldsymbol{v}_t \ \boldsymbol{q}_t \ \boldsymbol{a}_{bt} \ \boldsymbol{\omega}_{bt} \ \boldsymbol{g}_t \ \boldsymbol{\lambda}_t]^T \tag{4.55}$$

$$\boldsymbol{x}_n = [\boldsymbol{p}_n \ \boldsymbol{v}_n \ \boldsymbol{q}_n \ \boldsymbol{a}_{bn} \ \boldsymbol{\omega}_{bn} \ \boldsymbol{g}_n \ \boldsymbol{\lambda}_n]^T \tag{4.56}$$

$$\boldsymbol{x}_e = [\boldsymbol{p}_e \ \boldsymbol{v}_e \ \boldsymbol{\theta}_e \ \boldsymbol{a}_{be} \ \boldsymbol{\omega}_{be} \ \boldsymbol{g}_e \ \boldsymbol{\lambda}_e]^T . \tag{4.57}$$

The derivative with $\boldsymbol{\lambda}$ with respect to time can be easily derived since we assume the

scale factor is independent with time, therefore we have

$$\dot{\boldsymbol{\lambda}}_t = 0 \tag{4.58}$$
$$\dot{\boldsymbol{\lambda}}_n = 0 \tag{4.59}$$
$$\dot{\boldsymbol{\lambda}}_e = 0. \tag{4.60}$$

It is trivial to add map scale factors to system kinematic equations (e.g., Equation (4.8) to (4.13)) and the error state Jacobian (e.g., Equation (4.39) and (4.40)). The measurement function $h(\cdot)$ of estimated true state $\hat{\boldsymbol{x}}_t$ can be derived as

$$h(\boldsymbol{p}_t) = \boldsymbol{\lambda}_t \odot (\boldsymbol{p}_t - \boldsymbol{p}_{t0}) + \boldsymbol{p}_{t0} \tag{4.61}$$
$$h(\boldsymbol{v}_t) = \boldsymbol{v}_t \tag{4.62}$$
$$h(\boldsymbol{q}_t) = \boldsymbol{q}_t \tag{4.63}$$
$$h(\boldsymbol{a}_{bt}) = \boldsymbol{a}_{bt} \tag{4.64}$$
$$h(\boldsymbol{\omega}_{bt}) = \boldsymbol{\omega}_{bt} \tag{4.65}$$
$$h(\boldsymbol{g}_t) = \boldsymbol{g}_t \tag{4.66}$$
$$h(\boldsymbol{\lambda}_t) = \boldsymbol{\lambda}_t \tag{4.67}$$

where $\odot$ is point-wise vector multiplication and $\dot{\boldsymbol{p}_{t0}}$ is the camera/IMU position in reference frame, i.e., the position obtained from Homography of the first two keyframes. For simplification, we hereby assume the camera sensor and IMU sensor have identical position, i.e., $\dot{\boldsymbol{p}}_t$ is the estimated camera position in world frame. A solution to more complicated cases can be found in [26].

In Section 4.1, we have not given the explicit expression of $\mathbf{H}$ in Equation (4.42). $\mathbf{H}$ is defined as the Jacobian matrix of extrasensory measurement function $h(\cdot)$ with respect to the error state at nominal state $\boldsymbol{x}_n$, as $\boldsymbol{x}_n$ is the estimation of true state $\boldsymbol{x}_t$ here. By chain rule, $\mathbf{H}$ can be written as

$$\mathbf{H} \triangleq \frac{\partial h}{\partial \boldsymbol{x}_e}\Big|_{\boldsymbol{x}_n} = \frac{\partial h}{\partial \boldsymbol{x}_t}\Big|_{\boldsymbol{x}_n} \frac{\partial \boldsymbol{x}_t}{\partial \boldsymbol{x}_e}\Big|_{\boldsymbol{x}_n} = \mathbf{J}_H \mathbf{J}_{te}. \tag{4.68}$$

We have already given $\mathbf{J}_{te}$ in Equation (4.39), we then derive $\mathbf{J}_H$, which leads to

$$\mathbf{J}_H = \begin{bmatrix} \mathbf{J}_p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1}_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1}_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1}_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1}_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1}_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1}_3 \end{bmatrix}, \tag{4.69}$$

where $\mathbf{J}_p$ is a 3×3 matrix with diagonal being the three elements of $\boldsymbol{\lambda}_n$ respectively.

4.2.3 Keyframe-based Bundle Adjustment

At last, a keyframe-based bundle adjustment is used to further increase pose estimation accuracy.

A general bundle adjustment tries to optimize bunch of camera poses and 3D points together by minimizing the reprojection error between reprojected 3D points and predicted image points. Mathematically, we obtain our optimized camera poses vector $\boldsymbol{c}$ and 3D points vector $\boldsymbol{p}$ by

$$\{\boldsymbol{c}, \boldsymbol{p}\} = \arg\min_{\boldsymbol{c}_i, \boldsymbol{p}_j} \sum_{i=1}^{n} \sum_{j=1}^{m} Obj(CamProj(\boldsymbol{c}_i, \boldsymbol{p}_j), \mathbf{I}_{ij})^2, \tag{4.70}$$

where n and m are the number of camera poses and visible 3D points respectively, $\mathbf{I}_{ij}$ is the predicted 2D point position corresponding to j^{th} visible 3D point in i^{th} image frame, function $CamProj(\cdot)$ reprojects the visible 3D point from global frame into camera frame and $Obj(\cdot)$ is a general function (e.g., Euclidean distance) that measures the error between two 2D points.

As long as we assume our camera is pin-hole model, we can give function $CamProj(\cdot)$ as

$$CamProj(\boldsymbol{c}_i, \boldsymbol{p}_j) = K(\mathbf{R}\{\boldsymbol{c}_i\}\boldsymbol{p}_j) \tag{4.71}$$

$$K(\boldsymbol{P}_{\mathcal{C}}) = [u_0 \; v_0]^T + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} r \begin{bmatrix} \frac{P_x}{P_z} & \frac{P_y}{P_z} \end{bmatrix}^T, \tag{4.72}$$

where $\mathbf{R}\{\boldsymbol{c}_i\}$ is the rotation matrix corresponding to camera pose, function $K(\cdot)$ transforms a point P in camera frame $\mathcal{C}$ into 2D point in image plane. Parameters u_0, v_0 are principle point, f_u, f_v are focal length, and r is the distortion factor, where these parameters are obtained by the camera calibration.

After correction step, we employ such bundle adjustment (BA) step at each keyframe by inputting the nominal state camera poses and 3D points; we will apply an extra correction step using the BA results to further improve the estimation quality. In order to ensure the efficiency, we have ignored the oldest key frame after our key frame queue has been reached maximum number, we set this number to 20 in all our experiments. We have observed this number is suitable and keep our visual-inertial odometry runs in real-time, while improving the estimation accuracy at the same time.

Solutions to general bundle adjustment varies [49, 40, 24], we have mainly follow the work by [49] since it has been shown efficient for sparse bundle adjustment problem.

4.3 Visual-inertial Odometry Pipeline Summary

In this section we summarize our visual-inertial odometry (VIO) pipeline (see Figure 4-1). We also analyses the computational cost of our visual inertial odometry system

in this section.

An initialisation step is necessary for both nominal state and error state. After the system obtains the measurements (e.g., 3-vector from accelerometer, and 3-vector from gyroscope) from the IMU sensor, nominal state and error state are then predicted using Equation (4.26) to (4.37). The system updates corresponding covariance matrix using Equation (4.45).

If the camera gives an estimated camera pose in this turn, the system applies a *correction step* as we have described in Section 4.1.3. After injecting error state from nominal state, we obtain the estimated true state. System then reset the error state, and further improving the odometry estimation quality by applying a keyframe bundle adjustment (keyframe BA) if this camera frame is specified as key frame by visual odometry as we have discussed in Section 4.2.3. Finally the system outputs the estimated camera pose and starts the next round after having received IMU measurements.

In our ESKF framework, computational complexity remains constant since we only keep the current states and covariance, and the size of states and covariance matrix is fixed during estimation process. In order to verification, we have demonstrated an experiment (see Section 5.2.2) to show that computational time approximately remains unchanged when running a single IMU integration process using ESKF. As we have discussed in Section 2.2, the computational complexity of keyframed-based visual odometry is

$$O(m^2 \cdot n), \tag{4.73}$$

where m is the number of key frame, and n is the number of landmarks. We claim that our odometry can also be extended to large scale since the number of landmarks have no impact on our time consuming and this is in general the key factor of limiting the scalability of SLAM-like system. Moreover [37] have shown that it is more beneficial to obtain high estimation accuracy by increasing the number of landmarks than number of key frames.

We are not able to build an environmental map due to the time limitation of this master thesis. Although a few more map optimization steps are needed, the key frame BA we have proposed in Section 4.2.3 gives a direct result of optimized landmarks, which are the main components of mapping.

In summary, in this chapter we suggest an error-state Kalman filter (ESKF) based visual-inertial odometry. The system accurately estimates the camera-IMU platform pose by fusing the measurements from the visual-inertial sensors. Using a loosely-coupled approach, the system runs ESKF in a constant computational complexity and needs no special initialization steps, therefore it is suitable for mobile robot localization in real-time. This odometry can also be extended to an efficient and scalable SLAM system.

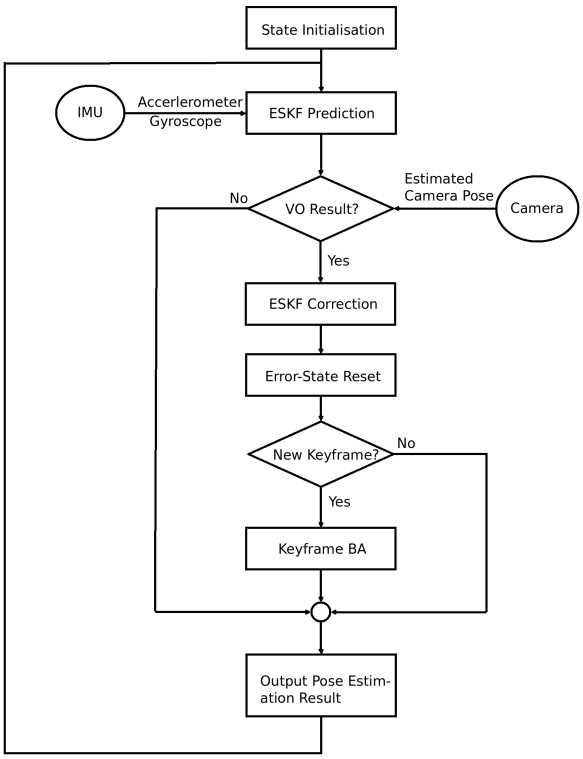


Figure 4-1: Pipeline of our visual-inertial odometry. Noted that measurement frequency from camera is 4 times lower than measurement frequency of IMU sensor in our experiment setting, therefore we do not have visual odometry (VO) result in some certain turn. Also whether this frame is keyframe is provided by VO.

Chapter 5

Experiments

In this chapter, we first introduce our synthetic dataset. This dataset contains IMU measurements and corresponding visual frames, it also provides the ground truth IMU-camera pose to be used in evaluation stage. We then perform several experiments to show that our visual-inertial odometry has good accuracy and low computational cost with different experimental settings.

5.1 Synthetic Dataset

We use a synthetic IMU-camera dataset in this master thesis because it has following advantages rather than real datasets,

- Noises in synthetic dataset is controllable. Though we have considered the noise model in our work, the types of noises in real data varies. Besides, denoising from the IMU sensor is beyond the content of this master thesis.
- Synchronization between IMU and camera is easily to be solved. Though we are able to obtain the output frequency of sensors from datasheet or introduction, it is non-trivial to align visual data and inertial data since the frequency of real sensors might be influenced by many environmental factors. In synthetic data, IMU and camera data is aligned automatically because their outputs are stable and fixed.
- Synthetic data is more flexible. We could generate the special cases, i.e., a pure rotation sequence. It is often hard to create such a sequence in the real platform.
- Synthetic data can provide accurate ground truth data **at each timestamp**. In real platform, a evaluation of position drift usually only happens at end point, e.g., measuring the drift by setting the start point as same as end point. Though a motion capture system is supposed tp provide ground truth data at each timestamp, accuracy of such system usually might be inadequate, and the cost is expensive.
- Numerous calibration steps are saved. In real equipment, it is quite usual to calibrate the device again after several uses. We can ignore the calibration processes in simulated platforms.

Name	Duration $[s]$	Distance $[m]$	Description
001	30.0	165.97	Pure movement without rotation
002	30.0	0.0	Pure rotation without translation
003	30.0	118.99	Random straight trajectory
004	120.0	257.71	Random circle trajectory

Table 5.1: Trajectories we create for experiments, the datasets are named after the corresponding trajectories. Note that **001** and **002** are ideal trajectories in order to evaluate the correctness and performance of ESKF IMU integration. The trajectory **003** and **004** try to simulate different types of real micro helicopter trajectories.

Also to best of our knowledge, there is no open sourced IMU-camera synthetic dataset until the writing of this thesis, therefore we decide to generate our own IMU-camera synthetic datasets.

Trajectory We start to generate synthetic visual-inertial data by first defining the trajectory of our IMU-camera platform. In VIOs the transformations between the camera sensor and IMU sensor is fixed and pre-calibrated, we assume the camera and IMU sensor share an identical pose in all our experiments for simplification. We have created four different trajectories shown in Table 5.1. Dataset **001** and Dataset **002** are used to evaluate the performance of ESKF IMU integration, therefore we apply ground-truth data in *correction step*. Dataset **003** and Dataset **004** both evaluate the performance of visual-inertial odometry except **004** has longer travelled distance and more general shape. There are several general assumptions we have made for simulated trajectories, which are

- We assume the initial position of trajectory as $(0, 0, 0)$ in global frame, and initial orientation as quaternion $(1, 0, 0, 0)$ which points at the positive z axis of global coordinate system.
- We assume the second derivative of position and orientation remains constant within two successive sensor samplings.
- We assume the second derivative of position and orientation obeys a normal distribution with additional white Gaussian noise.

Synthetic IMU data IMUSim [50] is the major tool to simulate IMU data with our customized trajectory. IMUSim is a powerful python-based open-sourced IMU simulation tool, which models a wide range of real-world environments with feasible external noises. To obtain more realistic IMU data, we set the output frequency of IMU as 100 [Hz], sensitivity of gyroscope to 1200 [deg/s] and sensitivity of accelerometer to 4 gravity. The IMU measurements are modelled with the additional white Gaussian noise. Overall we have 3-vector gyroscope readings, 3-vector accelerometer readings, 3-vector ground truth position, 4-vector ground truth orientation from IMUSim for single IMU sampling.

Dataset	fx	fy	cx	cy	d0	d1	d2	d3	d4
003	315.5	315.5	376.0	240.0	0.0	0.0	0.0	0.0	0.0
004	315.5	315.5	376.0	240.0	0.0	0.0	0.0	0.0	0.0

Table 5.2: Intrinsic matrix is only set for trajectory **003** and **004** since visual data is not required for trajectory **001** and **002**. We use the similar annotation for intrinsic matrix as in [38]. The parameter settings are similar with default ROS camera [33].

Synthetic visual data We simulate virtual scene by Blender [44]. We exploit high-frequent grass-like textures and sun lights to simulate the realistic outdoor environment. Thanks to Python interface of Blender, we can import our defined trajectory, and then render a video clip as our visual data set. Here we use linear interpolation for position, e.g., a position $\boldsymbol{p}$ between two successive ground truth positions $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t+1}$ can be computed as

$$\Delta t = \frac{t_p - t}{t_1 - t} \tag{5.1}$$

$$\boldsymbol{p} = \boldsymbol{p}_t + (\boldsymbol{p}_{t+1} - \boldsymbol{p}_t)\Delta t, \tag{5.2}$$

where t_p, t_1 and t is time when $\boldsymbol{p}$, $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t+1}$ have been measured. We use *Slerp* to interpolate between two quaternions $\boldsymbol{q}_t$ and $\boldsymbol{q}_{t+1}$ using Equation (5.1), we obtain the result $\boldsymbol{q}$ as

$$\boldsymbol{q} = \frac{\boldsymbol{q}_t \sin\left((1 - \Delta t)\theta\right) + \boldsymbol{q}_{t+1} \sin\left(\Delta t\theta\right)}{\sin\theta}, \tag{5.3}$$

where θ is the half angle between $\boldsymbol{q}_t$ and $\boldsymbol{q}_{t+1}$.

The resolution of image is 752×480, and intrinsic matrix of our virtual pin-hole camera is pre-defined, which is showed in Table 5.2. The output frequency is set to 25 [Hz], three times less than output frequency of the IMU sensor.

5.2 Experimental Results

5.2.1 Implementation Details

Our implementation consists of a visual odometry based on SVO [13] and a loosely-coupled ESKF framework. The goal of visual odometry is to track landmarks (for keyframe BA), estimate camera poses, and select keyframes as we discuss in Section 4.2. ESKF framework aims to integrate IMU measurements over timestamps, and fuse the visual results in correction step.

SVO [13] is a fast semi-direct monocular visual odometry. Instead of applying image features, the camera pose is estimated by minimizing photometric errors between successive images. SVO is very efficient because the high-cost feature extraction step has been skipped, besides a probabilistic mapping method has been used, therefore more reliable 3D landmarks can be obtained. Overall SVO provides a efficient and

robustness way to obtain camera poses, keyframes and 3D landmarks.

We implement C++ based ESKF framework as we explain in Section 4.1. Mathematical operations such as quaternion operations, and matrix operations are implemented with efficient C++ template linear algebra library Eigen [15]. In keyframe BA part, the basic BA solver is provided by Google Ceres Solver [3], we select the maximum number of keyframes to 20 for efficiency reasons, the BA step runs in parallel with ESKF. IMU measurements has been already aligned with visual measurements initially and manually (every four IMU data corresponds to one image frame).

All experiments are processed with a Intel Core i7 4700MQ laptop, and all codes related to this master thesis will be published on `https://github.com/OscarLiXi/MasterThesis` afterwards.

5.2.2 Experiment 1: IMU Integration

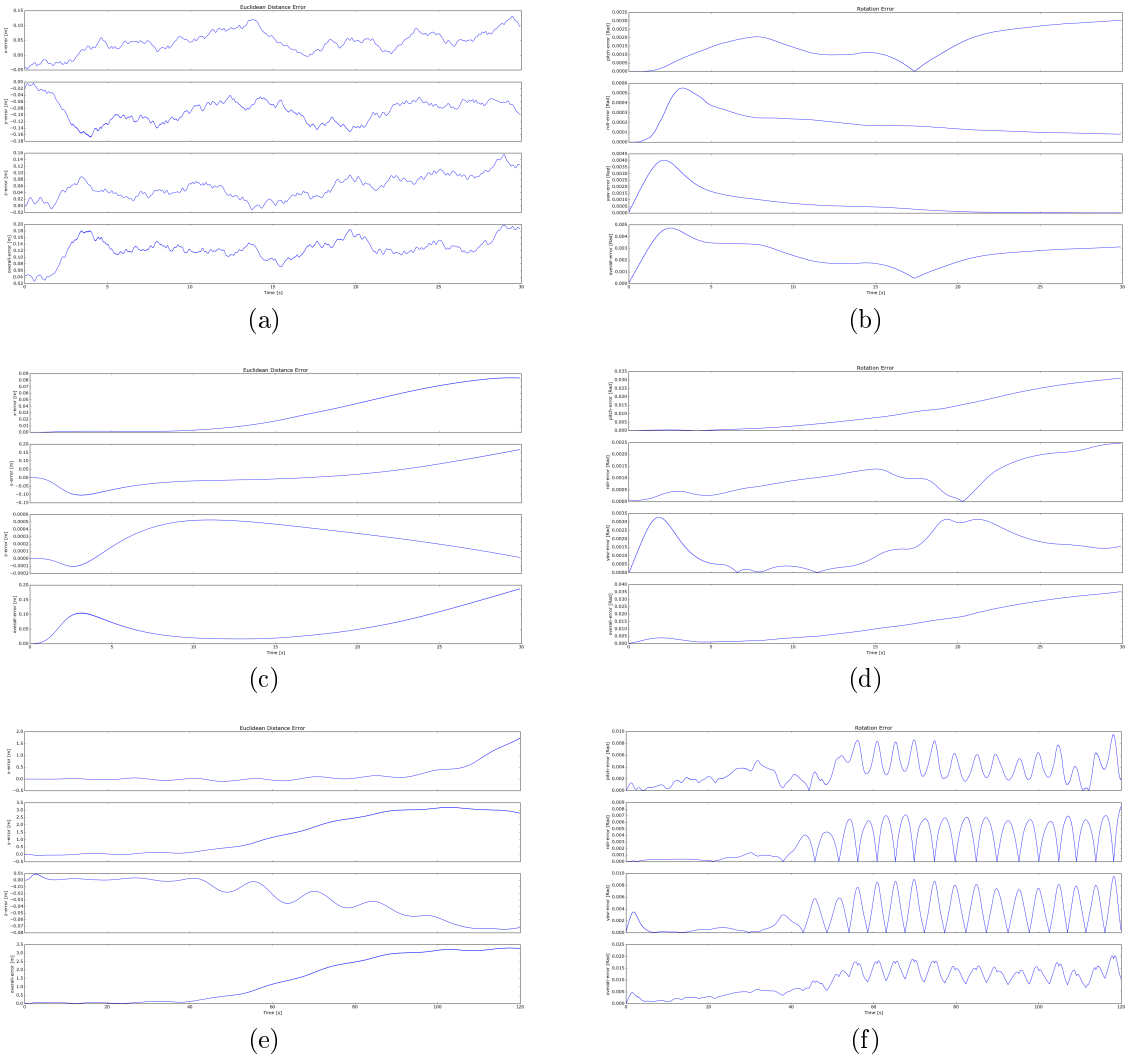
In this experiment, we examine the correctness and efficiency of single IMU integration (without extrasensory correction data). Since **001** and **002** are special cases that many VO system [8, 11] have failed, and dataset **004** is a general simulated micro helicopter trajectory, these three datasets are applied in this experiment to show the correctness of our IMU integration. Moreover we assume the VO provides us a very accurate camera pose estimation in correction step at a lower rate.

We have reported the results in Figure 5-1 and Table 5.3. For trajectory **001**, the average position drift is approximately 0.125 [m], which is less than 0.08% of total travelled distance (see Table 5.1). For trajectory **002**, the average attitude drift is approximately 0.0134 [rad], which is less than 0.8 [deg]. For more realistic and longer trajectory **004**, the average position drift is larger because error accumulation is inevitable, the average error is less than 0.55% of overall distance travelled; the results of average attitude error is also acceptable but has larger variance as we can see from Figure 5-1f. The time for processing a single IMU measurement does not grow as time goes by, which supports our claim that ESKF IMU integration runs under a constant time computational complexity. Furthermore, ESKF needs approximately 0.019 [ms] in a single run, therefore it runs in a real-time considering the output frequency of IMU is 100 [Hz].

We conclude from this experiment that our ESKF IMU integration can deal with normal and special cases, and it has a precise pose estimation results **if VO system has given a accurate correction**. In next few experiments we will abandon this assumption, and demonstrate some other comparisons. In addition, experimental results show that ESKF IMU runs in real-time, which the computational complexity of single IMU integration remains constant as we point out in Section 4.3.

5.2.3 Experiment 2: VIO Versus VO

In this experiment, we will compare our suggested VIO with single VO using same data sequences. As we have discussed before, correction data by visual sensor has a large impact on our final pose estimation since global translation and angle rotated



(a) (b) (c) (d) (e) (f)

Figure 5-1: Experiment 1: (a)(b) Position and attitude drift of Dataset **001** (c)(d) Position and attitude drift of Dataset **002** (e)(f) Position and attitude drift of Dataset **004**. Position error is measured using Euclidean distance, and attitude drift is measured by transforming quaternion back into Euler angle (see Appendix C) since it is more straightforward to evaluation. Ground truth data is provided by synthetic IMU data generator.

Dataset	APD [m]	AAD [Rad]	TPD [ms]
001	0.1251	0.0025	0.0184
002	0.0622	0.0134	0.0188
004	1.4319	0.0094	0.0187

Table 5.3: Experiment 1: APD (Average Position Drift) and AAD (Average Attitude Drift) is measured by averaging overall position and attitude drift. For TPD (Time-elapsed Per Data), we run the experiment 100 times to measure a overall elapsed time, then divide by number of experiments and number of IMU measurements to obtain the single data processing time.

around gravity vector (yaw) are unobservable for IMU. In loosely-coupled IMU integration, ESKF needs visual results because it is not aware of exact movement model. Here we propose the experiment 2 roughly under the pipeline in Section 4.3 except we exclude the keyframe BA procedure, which will be added as an extra correction shown in the experiment 3.

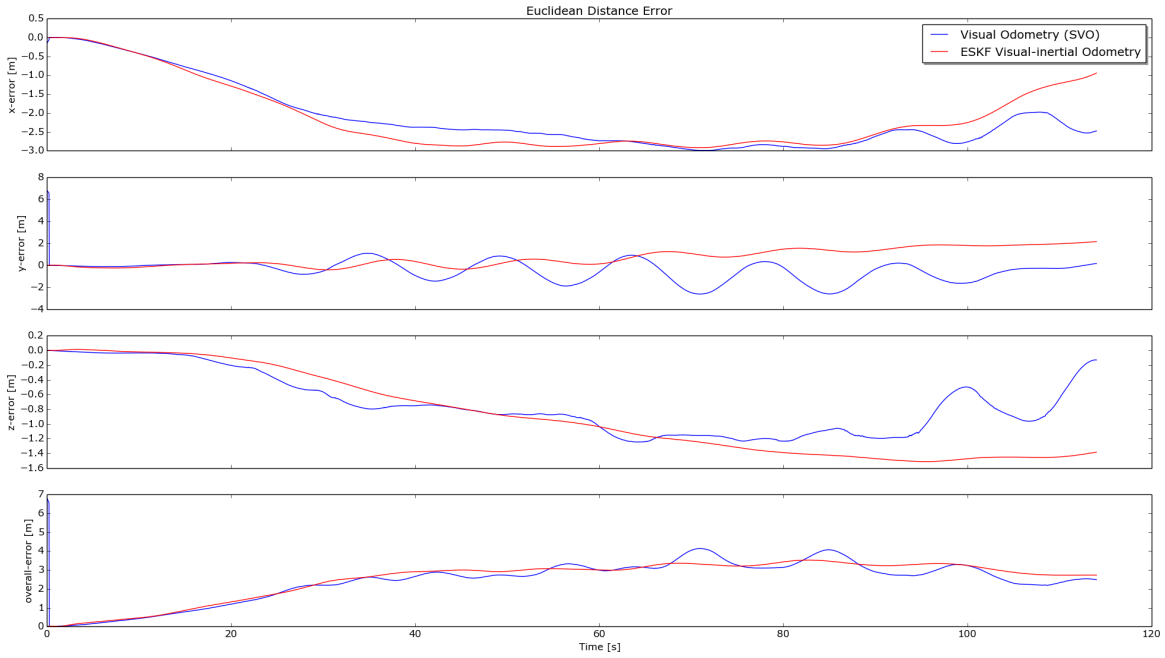
The results of experiment 2 on trajectory **004** have been shown in Figure 5-2. It is clear that VIO has lower variance than VO in translation though they have similar average position drift. VIO has given better results than VO regarding to the attitude drift. The reason is VIO gains much lower error than VO in two observable parameters (pitch and roll), and we can also see that VIO and VO report the similar error curve in yaw. These results are competitive among similar systems [29, 12] with similar overall travelled distance. We also evaluate the processing time using similar way as in Section 5.2.2, VIO finally has an average time per IMU data as 4.570 [ms], which processes the measurements in real-time.

We conclude that the VO has a huge positive impacts on 3-vector global translation and rotational angle around gravity vector. The estimated results by fusing visual data and IMU data has lower variance and higher accuracy than single visual odometry, and undoubtedly single IMU integration.

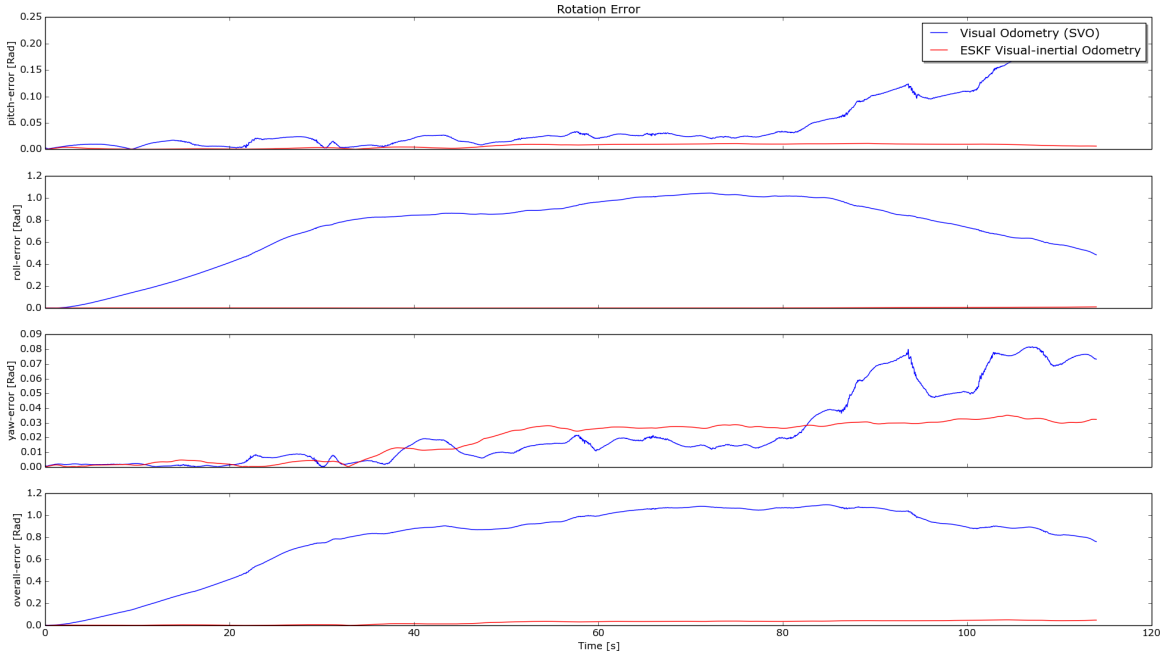
5.2.4 Experiment 3: Keyframe Bundle Adjustment

We further show the keyframe BA improves estimation quality by performing the experiment 3. In the experiment 3, we run our VIO on trajectory **004** with and without kerframe BA. Each BA step only happens when new keyframe is inserted, it will terminate after meeting the maximum iteration number (we set this number to 10) or a small threshold value. After the BA step, an extra correction step will be applied on error state using the results from keyframe BA.

In Figure 5-3, we report the results of VIO with BA and without BA. The red curve (VIO with keyframe BA) is more close to zero for most time, which shows that keyframe BA improves the camera pose estimation quality. Moreover blue curve reports an average position drift of 2.67 [m], where red curve reports 2.49 [m], which implies that VIO with keyframe BA have 10 % less position drift than VIO without BA on average. For attitude, VIO with keyframe BA also have less drift than VIO without keyframe BA, which former reports an average attitude drift of 0.028 [rad] and



(a)



(b)

Figure 5-2: Experiment 2: (a) Position drifts of VIO and VO. (b) Attitude drifts of VIO and VO. Both VIO and VO report an average position drifts less than 1% of overall travelled distance, however VIO performs regarding to the attitude drift, where VIO has 0.027 [rad] and VO has 0.787 [rad] average attitude drift.

	Time Consuming [ms]
ESKF Prediction	0.0146
Visual Odometry	4.4656
ESKF Correction	0.0422
Keyframe BA	0.2671
Total Visual-Inertial Odometry	4.7913

Table 5.4: Experiment 4: Break-up timing results.

Name	Parameter Name	Value
ESKF IMU	Nominal state integration	Euler
	Error state integration	Truncated Euler
VO	Max number of features per image	120
	Max number of keyframes	20
Keyframe BA	Max number of camera poses	20
	Max number of iterations	10

Table 5.5: Experiment 4: parameter settings in experiment 4. We hereby utilize the similar settings as we explained in experiment 1, 2, and 3.

the other is 0.029 [rad]. For processing time, keyframe BA increases approximately 0.5 [ms] for single IMU measurement on average, which leads to 5.08 [ms] per IMU measurement.

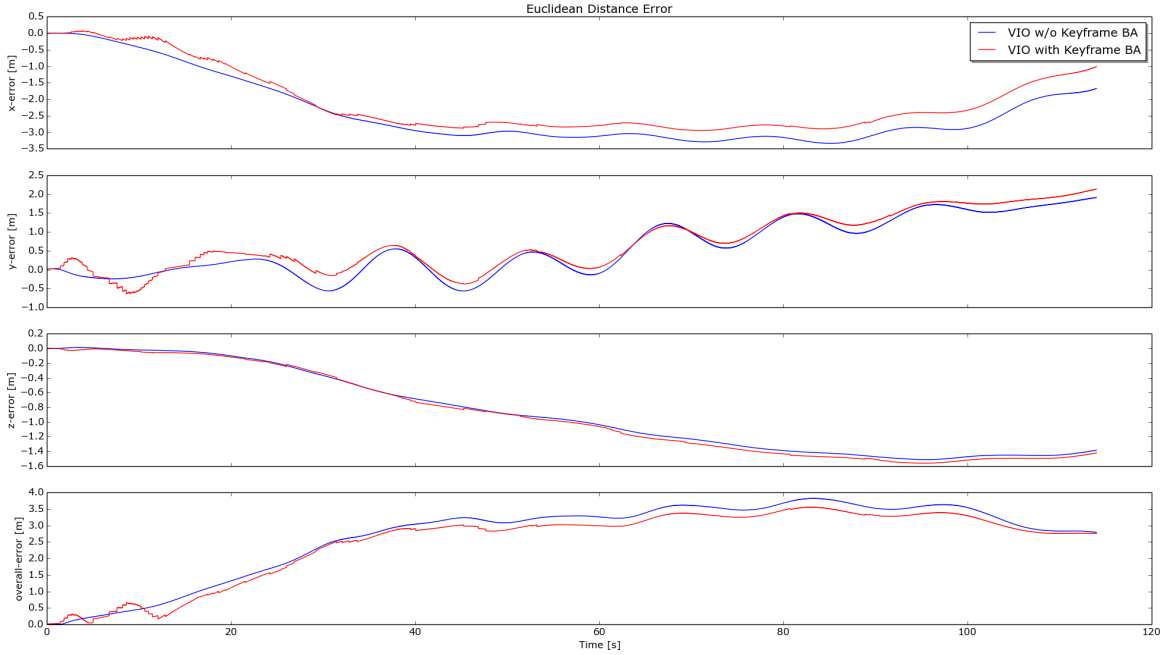
We conclude that keyframe BA as a motion optimization step improves the estimation quality in our case. Within 10 maximum iterations for a single BA turn, the whole system still runs in real-time.

5.2.5 Experiment 4: Runtime Evaluation

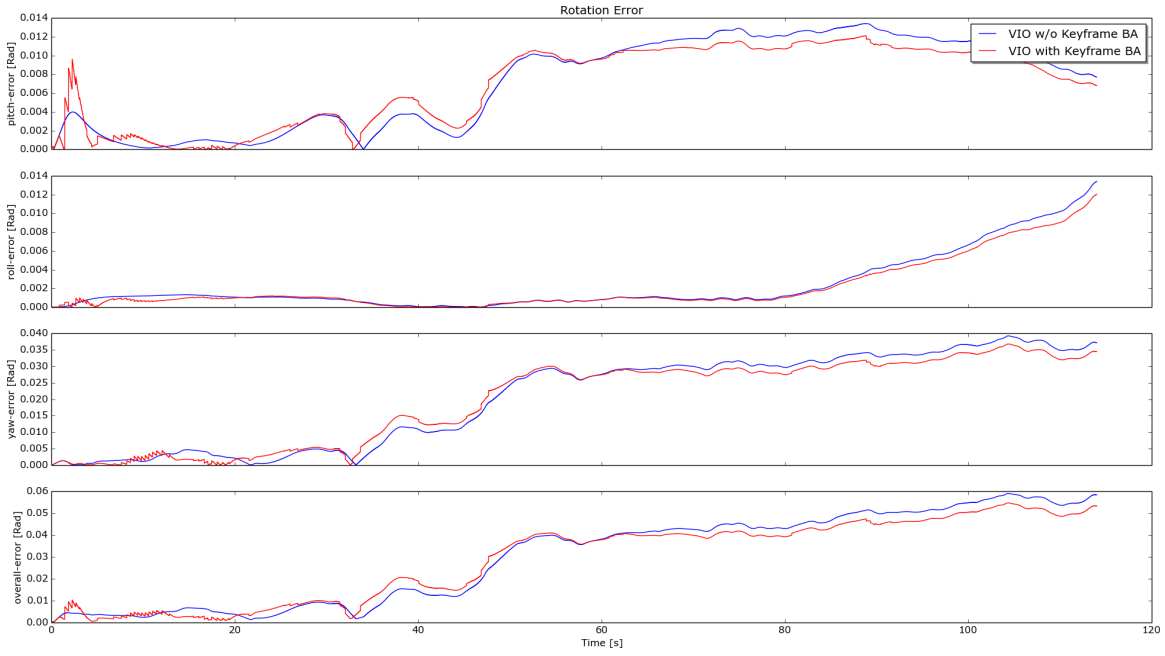
Table 5.4 shows a break-up of average time consumings for single IMU measurement to estimate camera motion in our synthetic datasets. In this experiment, we follow the parameter settings with former experiments, which explicitly shows in Table 5.5. We run experiments on Dataset **003** and Dataset **004** 100 times respectively in order to obtain more statistical results.

Though we have shown that our ESKF framework runs in constant time complexity, the whole VIO spends most of time on VO part. For VO part, we use *Fast* parameter settings as [13] suggests, except that we compromise the maximum keyframe number to 20 in order to obtain a more accurate pose estimations . The whole system runs smoothly in real-time with our virtual IMU sensor and camera settings, which needs 4.7913 [ms] to process single IMU measurement.

We conclude that the VIO we suggest in this master thesis have the ability to accomplish a real-time navigation task. It will further be significantly speed-up when more efficient and robust VO has been exploited.



(a)



(b)

Figure 5-3: Experiment 3: (a) Position drifts of VIO with and without BA. (b) Attitude drifts of VIO with and without BA. The red and blue curves show the errors for VIO with and without keyframe BA respectively.

5.3 Conclusion

We first explain the reasons and ways we generate synthetic datasets in this chapter. Then we demonstrate four experiments on our synthetic dataset. In experiment 1, we use the ground truth data in correction step, showing that if the extrasensory data gives us an accurate camera pose estimation, our visual-inertial odometry would have rather small position drift (less than 0.55% of overall distance travelled) and attitude drift (average 0.0134 [rad]). In experiment 2, we compare our visual-inertial odometry with state-of-art visual odometry *SVO*, the final results show that our method have less variation in position drift and large improvements (0.027 [rad] vs. 0.787 [rad]) in average attitude drift than SVO. In the third experiment, we further show that the keyframe BA we propose decreases the average position drifts and attitude drifts. We examine the break-up time consumptions in experiment 4, that our odometry runs in real-time with our virtual IMU ($\sim$ 100 [Hz]) and camera ($\sim$ 25 [Hz]).

Due to the time limits of this thesis, we are not available to perform more evaluations, e.g., using different visual odometries or applying our system on more realistic datasets. However the results we obtain in above experiments have shown that the keyframe-based visual-inertial odometry we suggest is robust and efficient .

Chapter 6

A Brief Introduction to Nonlinear Optimization-based Visual-Inertial Odometry

In the previous chapters, we introduce our loosely-coupled visual-inertial odometry using ESKF, and show our results are slightly better than single visual odometry. Generally, an ESKF update can usually be regarded as one iteration during optimization. Though we apply the bundle adjustment to further improve the estimation quality, the internal correlations among different sensors have been ignored. We hereby explore a more general approach, e.g., a nonlinear optimization way, to solve visual-inertial odometry problem in this chapter. Our presentations in this chapter are based on [21] and [12]. Note that we do not perform any experiments due to the time limitation of this master thesis, the experimental results might be included in future publications.

6.1 Cost Function of Visual-Inertial Odometry

Intuitively, in order to formulate our problem in non-linear optimization way, a cost function to minimize the visual error and IMU temporal error is required. This cost function is precisely the reprojection error and IMU error state we have introduced in Chapter 4.

We first define the set of true states up to the frame n as

$$\mathbf{X}_n \triangleq \{\boldsymbol{x}_t^i\}_{i \in \mathbf{K}_n}, \tag{6.1}$$

where $\{\boldsymbol{x}_t^i\}$ denotes the i^{th} frame of true states, and $\mathbf{K}_n$ represents the set of keyframes until frame n. Therefore our optimal set of true state $\mathbf{X}_n^\star$ is

$$\mathbf{X}_n^\star \triangleq \arg\min_{\mathbf{X}_n} \|e_v\|^2 + \|e_{imu}\|^2, \tag{6.2}$$

where e_v, e_{imu} are reprojection error, IMU temporal error until the frame n respec-

tively.

Reprojection error can written e_v by Equation (4.70) and Equation (4.71) as

$$\|e_v\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} \left\|CamProj(\boldsymbol{c}_i, \boldsymbol{p}_j) - \mathbf{I}_{ij}\right\|^2, \tag{6.3}$$

where n, m is the number of camera poses and visible 3D points respectively, and $\mathbf{I}_{ij}$ is the predicted 2D point position corresponding to j^{th} visible 3D point in i^{th} image keyframe.

For IMU temporal error, $\|e_{imu}\|^2$ can then be denoted via our error state $\boldsymbol{x}_e$ as

$$\|e_{imu}\|^2 = \sum_{i=1}^{n} \left\|\boldsymbol{x}_e^i\right\|^2, \tag{6.4}$$

where $\{\boldsymbol{x}_e^i\}$ denotes the i^{th} keyframe of error state. Under the assumption of zero-mean Gaussian noise, the cost function can further be written as

$$\mathbf{X}_n^\star \triangleq \arg\min_{\mathbf{X}_n} = \sum_{i=1}^{n}\sum_{j=1}^{m} \left\|CamProj(\boldsymbol{c}_i, \boldsymbol{p}_j) - \mathbf{I}_{ij}\right\|_{\boldsymbol{\Sigma}_v}^2 + \sum_{i=1}^{n} \left\|\boldsymbol{x}_e^i\right\|_{\boldsymbol{\Sigma}_i}^2, \tag{6.5}$$

where $\boldsymbol{\Sigma}_v$ and $\boldsymbol{\Sigma}_i$ are covariance matrices for reprojection error and IMU error.

6.2 Manifold Nonlinear Optimization

Given the cost function, it is non-trivial to solve it using general nonlinear optimization method (e.g., Gauss-Newton method) since rotational part of $\mathbf{X}_n$ belongs to a manifold. In this section, we introduce a standard approach for manifold optimization based on [2].

We first introduce the *exponential map*. A exponential map $exp(\cdot)$ of error state quaternion $\boldsymbol{q}_e$, which is very similar to Equation (3.23), is defined as

$$\boldsymbol{q}_e \triangleq exp(\boldsymbol{v}) = \mathrm{e}^{\boldsymbol{v}/2} = \begin{bmatrix} \cos\phi/2 \\ \boldsymbol{u}\sin\phi/2 \end{bmatrix}, \tag{6.6}$$

where $\boldsymbol{v}$ is the corresponding minimal axis-angle perturbation $\boldsymbol{v} = \phi\boldsymbol{u}$. Ar exponential map to transform between axis-angle perturbation and rotation matrix can be found in Equation (3.21). A exponential map maps minimal coordinates (e.g., angular error) into tangent space, which behaves as an Euclidean space locally. Due to this property, we can use the angular error in our error state to form a minimal axis-angle perturbation during the optimization process, and transform it back to quaternion afterwards using exponential map.

In summary, in each optimization iteration, we first perform general optimization approach on such minimal axis-angle perturbation since it behaves as Euclidean space locally. Then the current rotational guess is obtained via exponential map. By

using some properties of exponential map, one can decrease the optimization time significantly [12].

6.3 Disscussion and Conclusion

One problem of the optimization approach is that the optimized result is not easily calculated within a short time. Historical information has to be kept in order to apply a global optimization; new keyframes, landmarks and error states will be inserted to cost function continuously. Therefore solving such a cost function usually spends more time than our suggested ESKF framework. Though in [29], they solve it by only keep some of previous keyframes, but this is sub-optimal. However, in tightly-coupled visual-inertial SLAM, recent researches has partially addressed complexity problem by proposing *structureless approach* [29, 12] to efficiently eliminate all the landmarks (3D points) in optimization step. Also an IMU pre-integration technique [12] has been exploited to avoid recomputing relative pose transformations (via IMU measurements) between successive keyframes. These methods have decreased the optimization time significantly, thus non-linear optimization becomes feasible in the real-time VIO system. We aim to explore these approaches since the source codes of [12] and a nicely-structured visual-inertial dataset [6] have been released recently, unfortunately we are not able to finish coding and debugging until the writing of this thesis.

In this chapter, we introduce an approach to form a optimization cost function via similar notations in this master thesis. To conclude, though our suggested ESKF VIO runs in a constant time complexity, and obtain an acceptable estimation results, we could further improve our approach by focusing on internal connections between visual sensors and IMU sensors. Theoretically, if we can address the high computational complexity well, a more general approach, such as nonlinear optimization, might obtain more accurate results as similar time consumings as our suggested VIO.

Chapter 7

Summary, Discussion and Future Works

In this master thesis, we have suggested a robust sensory fusing framework, which can be applied on real-time mobile robot navigation. Visual data (e.g., images) can provide adequate information, however it is rather time-consuming to analysis; while the inertial sensor (IMU) provides accurate estimation of its observable variables but needs compensation in its unobservable parameters. We aim to improve the localization quality by fusing such multiple sensory data. This framework is lead by integrating IMU measurement from local frame to global frame, with additional extrasensory data (vision data) to complement unobservable parameters of IMU. We further explore a self-adapt map scale method and keyframe-based local bundle adjustment to increase the estimation accuracy.

In experimental parts, we demonstrate several experiments to provide evidences that our framework is robust and correct. First we examine our IMU integration method in regular and special trajectory independently, second we show that our framework is more robust and accurate than single visual odometry, at last we show that a keyframe-based local bundle adjustment to further increase the estimation accuracy, and whole system runs in real-time.

The advantages of this framework is that system does not need to keep visual landmarks and IMU measurement, therefore runs in a constant time complexity. Moreover it can be easily extended a large scaled scene. The drawback is that such a framework can be regarded as a single iteration of optimization step, hence the solution is sub-optimal. As we have discussed in the first few parts, the trade-off of computational complexity and estimation accuracy is inevitable, once we have used complete optimization solution, system will keep certain numbers of former states, hence it might be more time-consuming.

We briefly introduce our current interests in Chapter 6. We are supposed to show that our ESKF IMU integration framework can be transformed into a non-linear optimization solution easily. The cost function is formed by visual error (reprojection error) and IMU temporal error. By establishing such a cost function, one can solve it by performing a manifold optimization methods as we have introduced in Section 6.2. One thing left is to decrease the computational time in optimization step, such as land-

mark position elimination (structureless approach) in tightly-coupled VIO [29, 12], or pre-integration theory [12] to avoid repeatedly computing IMU integration. Another potential improvement is based on the work in [17], which provides a consistency analysis and modify some terms of traditional IMU integration formulations, and eventually obtain estimation results with less variations.

We are not able to build a real IMU-camera platform in this master thesis due to the lack of resources and time. A more rigorous denoise technique and more complicated noise model might have to be applied when using real data, which is also a possible working area in the future.

Appendix A

The Derivation of Error-state Kinematic Equations

We here derive Equation (4.21) and Equation (4.22) in Chapter 4.

In order to simplify our notation, we define

$$\boldsymbol{a}_n \triangleq \boldsymbol{a}_m - \boldsymbol{a}_{bn} \tag{A.1}$$
$$\boldsymbol{\omega}_n \triangleq \boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn} \tag{A.2}$$
$$\boldsymbol{a}_e \triangleq -\boldsymbol{a}_{be} - \boldsymbol{a}_n \tag{A.3}$$
$$\boldsymbol{\omega}_e \triangleq -\boldsymbol{\omega}_{be} - \boldsymbol{\omega}_n. \tag{A.4}$$

We derive Equation (4.21) by

$$\dot{\boldsymbol{v}}_e = \dot{\boldsymbol{v}}_t - \dot{\boldsymbol{v}}_n \tag{A.5}$$
$$= (\mathbf{R}_t\boldsymbol{a}_t + \boldsymbol{g}_t) - (\mathbf{R}_n\boldsymbol{a}_n + \boldsymbol{g}_n). \tag{A.6}$$

We then uses small signal approximation for R_t by Equation (3.29) and Equation (3.31), which leads to

$$\mathbf{R}_t = \mathbf{R}_n(\mathbf{1} + [\boldsymbol{\theta}_n]_\times) + O(\|\Delta\boldsymbol{\theta}_e\|^2). \tag{A.7}$$

We omit the second order of angular term and followed by Equation (4.15), we obtain

$$\dot{\boldsymbol{v}}_e = (\mathbf{R}_n(\mathbf{1} + [\boldsymbol{\theta}_n]_\times)\boldsymbol{a}_t + \boldsymbol{g}_t) - (\mathbf{R}_n\boldsymbol{a}_n + \boldsymbol{g}_n) \tag{A.8}$$
$$= \mathbf{R}_n(\mathbf{1} + [\boldsymbol{\theta}_n]_\times)(\boldsymbol{a}_n + \boldsymbol{a}_e) + \boldsymbol{g}_e. \tag{A.9}$$

By applying the property of skew-symmetric matrix $[\boldsymbol{a}]_\times \boldsymbol{b} = [\boldsymbol{b}]_\times \boldsymbol{a}$, and recalling (A.1), (A.2). We have

$$\dot{\boldsymbol{v}}_e = \mathbf{R}_n\left[\boldsymbol{a}_m - \boldsymbol{a}_{bn}\right]_\times - \mathbf{R}_n\boldsymbol{a}_{be} + \boldsymbol{g}_e - \mathbf{R}_n\boldsymbol{a}_n, \tag{A.10}$$

which is exactly Equation (4.21).

We then derive Equation (4.22) by

$$\dot{\boldsymbol{q}}_e = \frac{1}{2}(\boldsymbol{q}_e \otimes \boldsymbol{\omega}_t - \boldsymbol{\omega}_n \otimes \boldsymbol{q}_e), \tag{A.11}$$

and we have pure quaternion $\dot{\boldsymbol{\theta}}_e = 2\dot{\boldsymbol{q}}_e$. By expanding all terms in above equations, we have

$$\dot{\boldsymbol{\theta}}_e = -\left[\boldsymbol{\omega}_n\right]_\times \boldsymbol{\theta}_e + \boldsymbol{\omega}_e + O(\|\Delta\boldsymbol{\theta}_e\|^2). \tag{A.12}$$

By omitting all second-oder terms and recalling (A.3), (A.4), we obtain

$$\dot{\boldsymbol{\theta}}_e = \left[\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bn}\right]_\times - \boldsymbol{\omega}_{be} - \boldsymbol{\omega}_n, \tag{A.13}$$

which is precisely Equation (4.22).

Appendix B

Integration Methods

B.1 Runge-Kutta Numerical Integration Methods

Runge-Kutta methods aims to give a approximate solutions of ordinary differential equations, e.g., our time-derivative state $\dot{\boldsymbol{x}}$, which is

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}). \tag{B.1}$$

The solution give in Equation (4.14) - (4.19) by simplest version of Runge-Kutta methods, e.g., assuming $\dot{\boldsymbol{x}}$ over each time period Δt, which gives us

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + f(t_n, \boldsymbol{x}_n)\Delta t. \tag{B.2}$$

More complicated Runge-Kutta methods please refers to [48].

B.2 Closed-form Integration Methods

We first gives a clean version of Equation (4.22), which is

$$\dot{\boldsymbol{\theta}}_e = -\left[\boldsymbol{\omega}\right]_\times \boldsymbol{\theta}_e. \tag{B.3}$$

We update $\boldsymbol{\theta}_e$ by

$$\begin{aligned}
\boldsymbol{\theta}'_e &= \boldsymbol{\theta}_e + \dot{\boldsymbol{\theta}}_e \Delta t && \text{(B.4)}\\
&= \boldsymbol{\theta}_e - \left[\boldsymbol{\omega}\right]_\times \boldsymbol{\theta}_e \Delta t && \text{(B.5)}\\
&= \mathbf{R}\{-\boldsymbol{\omega}\Delta t\} && \text{(B.6)}\\
&= \mathbf{R}\{\boldsymbol{\omega}\Delta t\}^T, && \text{(B.7)}
\end{aligned}$$

which we apply Equation (3.29) and Equation (3.31) from Equation (B.5) to (B.6). This integration is a closed-form integration.

Appendix C

Conversion from Quaternions to Euler Angles

We introduce conversion from quaternions to Euler angles in this appendix. Though there are many conventions of Euler angles, we hereby specifically use so-called *Tait Bryan angles* to represent Euler angle. Tait Bryan angles is composed by Roll, Pitch and Yaw, which is the rotation angle around X-axis, Y-axis and Z-axis respectively. A quaternion $\boldsymbol{q}$ might be represented as

$$\boldsymbol{q} = [q_w \ q_x \ q_y \ q_z]^T . \tag{C.1}$$

Then Roll ϕ, Pitch θ and Yaw ψ can be obtained by

$$\phi = \arctan \frac{2(q_w q_x + q_y q_z)}{1 - 2(q_x^2 + q_y^2)} \tag{C.2}$$

$$\theta = \arcsin \left(2(q_w q_y - q_z q_x)\right) \tag{C.3}$$

$$\psi = \arctan \frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2)}, \tag{C.4}$$

which is precisely the attitude expression we use in experiment parts.

Bibliography

[1] Imu sensor https://www.vboxautomotive.co.uk/index.php/en/products/modules/inertial-measurement-unit, 2016. [Online; accessed 22-March-2016].

[2] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[3] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[4] Maxim A Batalin, Gaurav S Sukhatme, and Myron Hattig. Mobile robot navigation using a sensor network. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 636–641. IEEE, 2004.

[5] WG Breckenridge. Quaternions proposed standard conventions. *Jet Propulsion Laboratory, Pasadena, CA, Interoffice Memorandum IOM*, pages 343–79, 1999.

[6] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[7] Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4290–4297. IEEE, 2014.

[8] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE, 2003.

[9] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[10] Ethan Eade and Tom Drummond. Monocular slam as a graph of coalesced observations. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[11] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[12] 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 (RSS)*, 2015.

[13] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[14] G Guennebaud and B Jacob. The eigen c++ template library for linear algebra. *http:/eigen. tuxfamily. org*, 2010.

[15] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[16] William Rowan Hamilton. Ii. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(163):10–13, 1844.

[17] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. *Robotics, IEEE Transactions on*, 30(1):158–176, 2014.

[18] Roland Hess. *The essential Blender: guide to 3D creation with the open source suite Blender*. No Starch Press, 2007.

[19] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[20] Taehee Lee, Joongyou Shin, and Dongil Cho. Position estimation for mobile robot using in-plane 3-axis imu and active beacon. In *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pages 1956–1961. IEEE, 2009.

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

[22] Robert W Levi and Thomas Judd. Dead reckoning navigational system using accelerometer to measure foot impacts, December 10 1996. US Patent 5,583,776.

[23] Mingyang Li and Anastasios I Mourikis. Consistency of ekf-based visual-inertial odometry. *University of California Riverside, Tech. Rep*, 2011.

[24] Manolis IA Lourakis and Antonis A Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2, 2009.

[25] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[26] Simon Lynen, Markus W Achtelik, Steven Weiss, Maria Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3923–3929. IEEE, 2013.

[27] BL McNaughton, Lijiang Chen, and EJ Markus. âĂIJdead reckoning,âĂİ landmark learning, and the sense of direction: a neurophysiological and computational hypothesis. *Cognitive Neuroscience, Journal of*, 3(2):190–202, 1991.

[28] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M Newman, and Ian D Reid. A constant-time efficient stereo slam system. In *BMVC*, pages 1–11, 2009.

[29] Anastasios Mourikis, Stergios Roumeliotis, et al. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572. IEEE, 2007.

[30] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *Robotics, IEEE Transactions on*, 31(5):1147–1163, 2015.

[31] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[32] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652. IEEE, 2004.

[33] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[34] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[35] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[36] Joan Sola. Quaternion kinematics for the error-state kf. http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf, 2015. [Online; accessed 07-October-2015].

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

[38] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[39] Nikolas Trawny and Stergios I Roumeliotis. Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005, 2005.

[40] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmentâĂŤa modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 1999.

[41] JMP Van Waveren. From quaternion to matrix and back. *Id Software Inc*, 2005.

[42] Stephan M Weiss. *Vision based navigation for micro helicopters*. PhD thesis, Citeseer, 2012.

[43] Wikipedia. Rodrigues' rotation formula — wikipedia, the free encyclopedia, 2015. [Online; accessed 6-April-2016].

[44] Wikipedia. Blender (software) — wikipedia, the free encyclopedia, 2016. [Online; accessed 21-April-2016].

[45] Wikipedia. Global positioning system — wikipedia, the free encyclopedia, 2016. [Online; accessed 24-March-2016].

[46] Wikipedia. Homography — wikipedia, the free encyclopedia, 2016. [Online; accessed 29-May-2016].

[47] Wikipedia. Robotics — wikipedia, the free encyclopedia, 2016. [Online; accessed 22-March-2016].

[48] Wikipedia. Runge kutta methods — wikipedia, the free encyclopedia, 2016. [Online; accessed 11-April-2016].

[49] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064. IEEE, 2011.

[50] Alexander D Young, Martin J Ling, and Damal K Arvind. Imusim: A simulation environment for inertial sensing algorithm design and evaluation. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 199–210. IEEE, 2011.