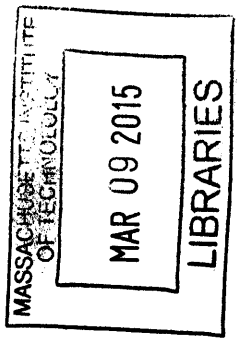
Visual-Inertial Odometry with Depth Sensing Using a Multi-State Constraint Kalman Filter

by

Marissa N. Galfond

B.S. Aerospace Engineering
University of Maryland, College Park, 2012



Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Signature redacted

Author .....
Department of Aeronautics and Astronautics
May 22, 2014

Signature redacted

Certified by .....
Paul A. DeBitetto, Ph.D.
Senior Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc.

Signature redacted

Thesis Supervisor

Accepted by .....
Paulo C. Lozano, Ph.D.
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee
Thesis Supervisor

Visual-Inertial Odometry with Depth Sensing Using a Multi-State Constraint Kalman Filter

by

Marissa N. Galfond

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The goal of visual inertial odometry (VIO) is to estimate a moving vehicle's trajectory using inertial measurements and observations, obtained by a camera, of naturally occurring point features. One existing VIO estimation algorithm for use with a monocular system, is the multi-state constraint Kalman filter (MSCKF), proposed by Mourikis and Li [34, 29]. The way the MSCKF uses feature measurements drastically improves its performance, in terms of consistency, observability, computational complexity and accuracy, compared to other VIO algorithms [29]. For this reason, the MSCKF is chosen as the basis for the estimation algorithm presented in this thesis.

A VIO estimation algorithm for a system consisting of an IMU, a monocular camera and a depth sensor is presented in this thesis. The addition of the depth sensor to the monocular camera system produces three-dimensional feature locations rather than two-dimensional locations. Therefore, the MSCKF algorithm is extended to use the extra information. This is accomplished using a model proposed by Dryanovski et al. that estimates the 3D location and uncertainty of each feature observation by approximating it as a multivariate Gaussian distribution [11]. The extended MSCKF algorithm is presented and its performance is compared to the original MSCKF algorithm using real-world data obtained by flying a custom-built quadrotor in an indoor office environment.

Thesis Supervisor: Paul A. DeBitetto, Ph.D.
Title: Senior Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc.

Thesis Supervisor: Paulo C. Lozano, Ph.D.
Title: Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Acknowledgments

The past two years have rewarded me with exciting research, valuable education and, most importantly, the chance to work alongside many remarkable people. I would like to thank Dr. Paul DeBitetto, Dr. Richard Stoner and the Draper Fellowship program for providing me with this opportunity. I am also greatly appreciative of Paulo Lozano for his guidance throughout my time in the graduate program.

I am grateful for the support I received from my group at Draper. I especially want to thank Andrew Reiter, Tim Dyer and Rob Truax for their invaluable help with this research. My friends and fellow labmates, Ben Arneberg, Matt Graham, Ted Steiner, Krish Kotru and many others have all been extremely helpful with research and this thesis preparation; plus, our afternoon coffee breaks and occasional trips to watch Champions League games have gotten me through the long days in the lab.

I want to thank my family, Leslie, Eric, Brian and Nikki, for helping take my mind off work during my occasional, and always eventful, visits home. Finally, I am truly thankful for the love and support from my parents. Mom, all the surprises you send me always make my day. Dad, I promise to get you the fastest motorized scooter to zip around in- after all, you can't put a price on happiness! I would not be where I am without the encouragement I have received from both of you over the years.

Assignment

Draper Laboratory Report Number T-1784

In consideration of the research opportunity and permission to prepare this thesis by and at the Charles Stark Draper Laboratory, Inc., the author hereby assigns copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts and the Massachusetts Institute of Technology. The author hereby grants to MIT and Draper Laboratory permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Marissa N. Galfond

Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Motivation

Unmanned aerial vehicles and mobile robots are commonly used in search and rescue missions and for exploration of unknown environments. Achieving robust pose estimation and navigation of these vehicles is a challenging problem, especially in GPS denied environments. Vision has been used for obstacle detection and navigation of vehicles and robots for many years [36]. Cameras have the advantages of low cost, weight and power consumption [34]; however, vision also has significant latency, computational complexity and potential robustness issues depending on the environment [19]. These characteristics make vision a natural supplement to inertial measurement units (IMUs) in aiding navigation, especially in GPS denied environments [34, 36, 19]. Visual-inertial odomerty (VIO) is the process of estimating a vehicle's trajectory using both inertial measurements and feature observations from cameras. There are several different VIO methods; the choice of method for a particular system depends on computational constraints, performance requirements and the available sensors.

The goal of this thesis is to develop and implement a VIO system on a quadrotor equipped with a monocular camera, an IMU and a depth sensor. The multi-state constraint Kalman filter (MSCKF) [34, 29], a sliding window VIO algorithm for real-time vision-aided inertial navigation, is the basis for the VIO method developed in this thesis. The main contributions of this thesis are extending the MSCKF to use 3D

feature measurements and presenting a feature location uncertainty model. Finally, this thesis analyzes the performance of the extended MSCKF through real-world experiments of navigating a quadrotor.

1.2 Literature Review

There are several visual-inertial odometry (VIO) methods in the literature that fuse visual and inertial measurements in different ways. These methods can generally be described as either "loosely coupled," in which IMU measurements and vision measurements are processed separately (i.e. each sensory system produces its own state estimate), or "tightly coupled", in which the vision measurements and inertial measurements are directly fused and the state is jointly estimated. Furthermore, tightly coupled methods can be divided into iterative minimization methods or Extended Kalman Filter (EKF)-based methods.

EKF-based VIO methods are less computationally expensive and, typically, less accurate than iterative minimization methods. However, the multi-state constraint Kalman filter (MSCKF) presented in [29] is an EKF-based method that is proven to be more accurate and less computationally expensive than iterative minimization methods. Thus, this thesis will focus on EKF-based VIO methods, and specifically on the MSCKF. The remainder of this section discusses the aforementioned VIO methods.

1.2.1 Loosely Coupled VIO

Loosely coupled VIO methods process IMU measurements and vision measurements separately. These methods are the most computationally efficient; however, when IMU measurements are processed separately from vision, the state estimates are not optimal [29]. Loosely coupled methods are commonly used in systems with computational limitations, such as micro air vehicles (MAVs) [5, 7].

Diel et al. present a loosely coupled VIO method that treats visual measurements as stochastic constraints on the camera position [9]. The method uses two camera

poses and a jointly observed feature to define a plane (i.e. an epipolar constraint.) Out-of-plane violations of this epipolar constraint are converted to residuals that are fed into a linear Kalman filter. Since the in-plane information is disregarded, this filter does not use all the feature information. The advantage of this loosely coupled filtering scheme is that computation and memory requirements scale linearly with the number of features.

Roumeliotis et al. obtain relative pose measurements of a vehicle using two-frame feature based motion estimation and then augment the information with inertial measurements [42]. In the two-frame feature based motion estimation, features are tracked between two images and the motion of the camera is estimated by solving for the rigid transformation that transforms the features from the first image into the features from the second image. The advantage of using only two image frames to estimate vehicle pose is increased computational efficiency. However, if a feature is seen in multiple images, additional constraints between multiple poses can be made [34]. Therefore, by only using two of the image frames to estimate motion, information from the remaining images is disregarded and motion is only sub-optimally estimated.

Konolige et al. develop a system that fuses visual odometry and IMU data using an EKF [24]. In this system, each sensory system produces a pose estimate which gets fused with the other pose estimates in an EKF. The pose estimate is sub-optimal since the existing cross-corrolations between internal states of the different devices are unused; however, the RMS error of the vehicle trajectory was reduced by an order of magnitude when VO was loosely coupled with the IMU measurements.

A more recent work [32] fuses stereo camera sensor data with IMUs and leg odometry on a dynamic legged vehicle. In this system, stereo visual odometry estimates vehicle translation and orientation by minimizing a non-linear cost function. The motivation for fusion of the IMU and leg odometry data with the VO pose estimate (accomplished using an EKF) is to reduce bias introduced by linearization errors when minimizing the cost function. The loose coupling of the IMU and VO sufficiently reduced drift in the vehicle position estimate.

Semi-Tightly Coupled VIO

Vision data and IMU data are processed separately, except for scale-factor estimation in [49] and [13]. The scale factor estimation uses both IMU and camera measurements; thus these algorithms can be classified as semi-tightly coupled. Observability analysis of the semi-tightly coupled system, in which IMU data and the map scale-factor are jointly estimated in an EKF, is presented in [49]. The analysis proves the theoretical observability of roll, pitch and sensor calibration states (IMU biases and camera-to-IMU transformation) of the vehicle. The map scale, and therefore, speed of the vehicle, are also observable in the system; this is crucial since [49] and [13], aim to robustly control the speed of an MAV in real time.

Real-time, onboard state estimation of an MAV using a single camera and an IMU is demonstrated in [49]. Similarly, a low-cost quadrotor equipped with an IMU and two cameras, developed by [13] achieved autonomous navigation. The limited computational platforms lead both [49] and [13] to tailor parallel tracking and mapping (PTAM)[1] to their systems to process vision measurements. The map-scale estimation, using both vision and IMU data, is more accurate in these algorithms than in loosely coupled VIO methods. Furthermore, the vehicle speed estimation is sufficiently accurate to robustly control the MAV in both semi-tightly coupled systems.

The main advantage of using loosely coupled (and semi-tightly coupled) VIO methods is the increased computational efficiency. In computationally constrained systems, such as [9, 42], the computational efficiency of loosely coupled VIO methods outweighs the disadvantage, namely, the sub-optimal state estimate. Additionally, depending on the computational constraints and the performance requirements of a system, loosely coupled VIO methods may be sufficient. For example, in the systems presented in [24] and [32], loosely coupling vision and IMU data was sufficient in improving long term pose estimation by reducing drift. The semi-tightly coupled systems, [49] and [13], achieve more accurate map-scale estimation and are still

[1]PTAM is presented in [21] to track a hand-held camera in an small, unknown environment by splitting tracking and mapping into two separate tasks. One thread is tasked with robustly tracking erratic hand-held motion, while the other thread creates a 3D map of point features from previously observed video frames.

computationally inexpensive. The goal of this thesis, however, is to develop a more accurate, tightly coupled VIO method that is still computationally inexpensive enough to run onboard on a quadrotor. The MSCKF, is ultimately used as the basis for the VIO method presented in this thesis.

1.2.2 Tightly Coupled VIO

Tightly coupled VIO methods directly fuse inertial and visual data; thereby obtaining more accurate state estimates but at a higher computational cost than loosely coupled schemes. Tightly coupled methods are further classified as either EKF-based methods or iterative minimization methods [29]. EKF-based VIO methods include EKF-SLAM algorithms, sliding window algorithms and hybrid algorithms. Iterative minimization methods more effectively deal with non-linearities in measurement models because re-linearzation is performed at each iteration; thus, they are typically more accurate than EKF-based methods, but come at a higher computational cost. Iterative minimization methods and EKF-based algorithms (including EKF-SLAM, sliding window and hybrid algorithms) are discussed in the following sections.

EKF-SLAM Algorithms

EKF-SLAM algorithms are tightly coupled EKF-based VIO estimators that jointly estimate the current camera pose and feature locations [29]. Since the feature locations are stored in the state vector, the computational cost of EKF-SLAM algorithms scales with the number of features in the state vector cubed [47]. In typical environments, large numbers of features are visible, causing the cost of an EKF-SLAM algorithm to be unacceptably high [29]. Therefore, in order to keep the computational cost of the algorithm bounded, features are usually removed from the state vector when they leave the camera's field of view [29].

Jones and Soatto integrate visual and inertial sensors using a standard EKF-SLAM framework for ego-motion estimation, localization and mapping [19]. They also address camera-IMU calibration, stabilization of the estimation process and handling

failure modes from each sensing modality. Other EKF-SLAM algorithms are presented by Kleinert and Schleith, where a camera-IMU system capable of real-time autonomous navigation and map building is developed [22], and by Pinies et al., who augment the algorithm presented by Civera et al.[2] with inertial measurements from a low-cost IMU in order to improve the estimated camera trajectory, map scale estimation and feature localization [38].

Since EKF-SLAM based algorithms and the MSCKF are both EKF-based estimators, they use the same measurement information [29]. Furthermore, Li and Mourikis prove that EKF-SLAM algorithms and the MSCKF would yield the same, optimal estimates for IMU pose if the systems were linear-Gaussian [29]. However, the actual measurement models are nonlinear, which allows the MSCKF to outperform EKF-SLAM algorithms in terms of accuracy. This is mainly due to the way feature measurements are used in the MSCKF, and this will be discussed in more detail when these algorithms are extensively compared in section 3.3. Additionally, the MSCKF's computational cost per time step scales linearly, rather than cubicly (as do EKF-SLAM algorithms), with the number of features [29]. Since the MSCKF is more accurate and computationally efficient than EKF-SLAM algorithms, it is the basis for the VIO algorithm presented in this thesis.

Iterative minimization methods

EKF-based estimators are susceptible to a gradual buildup of measurements linearization errors, which in VIO problems, causes trajectory estimation errors to continuously accumulate [10]. Iterative minimization methods have been developed to better handle such nonlinearities. In these methods, the filter maintains a sliding window of states, comprised of camera positions and feature locations. In order to obtain estimates of all the states, a nonlinear cost function consisting of feature location error terms and camera position error terms is minimized using an iterative mini-

[2]Civera et al. use standard monocular SLAM framework for real-time motion estimation and mapping with a hand-held monocular camera [8]. Through the use of an inverse depth parametrization, in which features are represented as a 6-D state vector, estimates of camera orientation are significantly improved and camera estimation jitter is reduced.

mization technique (i.e. iterative Gauss-Newton minimization) [10]. Because of the relinearization at each iteration, these methods have higher computational cost than most EKF-based methods.

The iterative minimization method for VIO used in [23] extracts feature locations from a small number of image frames. This sliding window of image frames, along with IMU data, is the input to a nonlinear least squares optimization to estimate the vehicle trajectory. In order to achieve real-time operation, the nonlinear least squares problem must be solved incrementally as described in [23]. An iterative minimization algorithm, presented in [10], tracks the motion of a robot by iteratively re-linearizing robot-to-landmark measurements and odometry measurements over a window of poses. In order to maintain bounded computational complexity for real-time operation, this algorithm is forced to marginalize out (i.e. remove) older poses from the state vector. Similarly, iterative minimization on a window of poses within a filter is used in [31]. As new poses are added and past observations are removed, the current window of states being estimated is independent from the previous estimates; thereby isolating errors in the filter to the region where they occur. The iterative minimization methods [23, 10, 31] run in real time and usually are more accurate than typical EKF-based methods.

The iterative minimization method presented by Sibley et al. is a sliding window bundle adjustment method [45]. Gauss-Newton minimization is used to update state estimates each time a new image is recorded; then, in order to maintain a constant computational cost, older states are marginalized out. An extensive comparison between this method [45] and the MSCKF is given in [29]. It is shown that the MSCKF algorithm achieves better consistency and accuracy even though the iterative minimization method had approximately fives time longer computation time. Even though the iterative minimization method better approximated the nonlinear measurement models by relinearizing at each time step, the MSCKF, which uses a linear model with appropriate observability properties,[3] yielded better results [29]. Since the MSCKF is an EKF-based estimator that is proven to achieve more accurate results than the

[3]The importance of correct observability properties of the MSCKF is discussed in section 3.3

best iterative minimization methods at a lower computational cost, the focus of this thesis will be EKF-based VIO algorithms and, specifically the MSCKF, which is the basis for the VIO algorithm presented in chapter 4 of this thesis.

MSCKF Algorithm

The multi-state constraint Kalman filter (MSCKF), a "sliding window" EKF-based VIO algorithm for real-time vision-aided inertial navigation, is presented in [34, 29]. The MSCKF is classified as "sliding window" because it maintains a sliding window of camera poses in the filter state and use feature measurements to impose probabilistic constraints on the camera poses [29].

This MSCKF differs from standard EKF-SLAM algorithms in the way that feature observations are used. In the MSCKF, a feature is tracked until it leaves the field of view of the camera; then all the feature measurements are used to form probabilistic constraints on the camera poses from which the feature was observed [29]. These constraints form the residuals used for EKF updates and are independent from the errors in feature locations. The MSCKF is able to use each feature measurement by expressing its information in a residual instead of including the feature location in the state vector. In contrast, EKF-SLAM algorithms inherently make assumptions that the probability distribution function of the feature's positions is Gaussian; this results in less accurate state estimates [29].

The MSCKF has two characteristics crucial for accurate state estimation: consistency and correct observability properties [29]. While EKF-SLAM algorithms (and iterative minimization methods) can be consistent, they do not possess correct observability properties [29] and the result is a decrease in accuracy. Both characteristics will be discussed in section 3.3. Finally, the computational complexity of the MSCKF is linear in the number of features [34]. This makes it more computationally efficient than EKF-SLAM algorithms and iterative minimization methods.

The accuracy and computational efficiency of the MSCKF make it the most suitable VIO method for implementation on a quadrotor. The MSCKF, as originally proposed, is implemented on a system with only a monocular camera and an IMU.

These sensors only provide two-dimensional feature locations. The purpose of this thesis is to extend the MSCKF to use three-dimensional feature measurements, which are obtained with the additional use of a depth camera. This thesis also presents a feature depth uncertainty model to use with the extended MSCKF.

Hybrids of the MSCKF and EKF-SLAM estimators

A hybrid estimator that integrates the MSCKF with EKF-SLAM is designed in [27]. This is accomplished by choosing the size of the sliding window, m. If a feature is lost after fewer than m frames, it is processed using MSCKF. Otherwise, the feature is initialized into the state vector and used for SLAM. The advantage of this algorithm, compared to the MSCKF and EKF-SLAM algorithms individually, is the improvement in computational efficiency. In simulations, the hybrid estimator has approximately half the runtime compared to each method. Real-world data shows the hybrid estimator is only slightly less accurate than the MSCKF (the hybrid still out performs the EKF-SLAM estimator in terms of accuracy). Another work, [50], uses the MSCKF as a basis filter, but only includes a pose in the state vector if it has a significant baseline from a previously stored frame. Additionally, the system keeps a small number of persistent features in the state vector. Similarly to the hybrid estimator presented in [27], the estimator in [50] sacrifices accuracy for improved computational efficiency. Thus, if processing power is limited, a hybrid estimator can be implemented with only a slight decrease in accuracy compared to the MSCKF.

1.2.3 Summary of VIO Methods

VIO methods can be generally classified as loosely coupled or tightly coupled. Loosely coupled VIO methods process vision measurements and inertial measurements separately; because of this, they are the most computationally efficient. Tightly coupled methods which, process vision measurements and inertial measurements together, are more accurate and are more relevant to this thesis.

EKF-based methods and iterative minimization methods are types of tightly cou-

pled VIO methods. EKF based methods are either EKF-SLAM algorithms, or, like the MSCKF, sliding window algorithms. Typically, iterative minimization methods are the most accurate and computationally expensive types of tightly coupled VIO methods due to the way in which they deal with measurement nonlinearities. The MSCKF, however, is more accurate and computationally efficient than the best iterative minimization methods and EKF-SLAM algorithms. These characteristics make the MSCKF the most suitable VIO method to implement on a quadrotor. As was previously mentioned, the purpose of this thesis is to extend the MSCKF to use three-dimensional (rather than two-dimensional) feature measurements. A feature depth uncertainty model is also presented to use with the extended MSCKF.

1.3 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the visual odometry problem and briefly discusses its key components. Additionally, SLAM will be introduced and the general EKF-SLAM solution, a framework for EKF-based VIO, will be summarized. Chapter 3 presents and compares two EKF-based VIO algorithms: Standard EKF-SLAM and the MSCKF. Chapter 4 explains how the MSCKF is extended to use 3D feature measurements and presents a feature depth uncertainty model. Chapter 5 describes the implementation of the extended MSCKF for a quadrotor and contains experimental results and analysis of the extended MSCKF. Finally, conclusions and future work are discussed in Chapter 6.

Chapter 2

Background

2.1 Visual Odometry

Visual odometry (VO) is a process of estimating egomotion, movement of a vehicle in a static environment, using only the input of a camera (or cameras) attached to the vehicle. In the early 1980's, Hans Moravec first introduced the problem of estimating egomotion from visual input and tested his work on a planetary rover equipped with a single camera sliding on a rail [33]. The term "visual odometry" was coined by David Nister in 2004 in a landmark paper describing a real-time method for estimating the motion of a vehicle from video sequences in a previously unknown environment. The system consists of a feature tracker, in which point features are matched between pairs of frames, producing feature tracks. The feature tracks are then used to estimate camera motion using a "geometric hypothesize-and-test architecture" [36].

Visual odometry can be classified as either stereo or monocular. In stereo visual odometry, the use of two cameras, or the ability to obtain two images from the same vehicle location, enables direct measurement of the relative 3-D position of features at every vehicle location. Alternatively, in monocular visual odometry, only a single camera is used; thus, only bearing information is available. In this case, vehicle motion can only be estimated up to a scale factor. This means that only the relative scale between two image frames can be estimated. However, direct measurements, such as the size of a landmark, motion constraints, or integration with other sensors

allow the determination of absolute scale for monocular camera systems. Monocular methods are important because, when the distance from the cameras to the scene is much greater than the the distance between the two cameras, stereo vision becomes ineffective and monocular methods must be used instead [43].

Similarly to inertial measurement units (IMUs), visual odometry functions incrementally by dead reckoning and accumulates error over time [36]. However, VO is not affected by global positioning system (GPS) dropouts, wheel slip or other adverse conditions; making it a natural supplement to many navigation sensors, such as GPS, IMUs and laser range scanners [36, 43].

2.1.1 Overview of the VO Problem

As a vehicle moves through an environment, the images produced by the onboard camera(s) change accordingly. Visual odometry incrementally estimates the pose of the vehicle based on how the images change. Therefore, in order for VO to work effectively, sufficient illumination is required in a static environment with rich texture [43]. The illumination and texture allow multiple features to be extracted from image frames. A static environment is important because VO operates by tracking the change in feature locations from frame to frame. If a feature has actually moved within the environment, treating it as a stationary point of reference will cause an erroneous estimate of vehicle motion [43]. Additionally, VO requires consecutive frames to have sufficient scene overlap [43]. This allows features to be seen and tracked over many different image frames. With these requirements, an illustration of the VO problem is shown in figure 2-1.

As seen in figure 2-1, image I_{k-1} is produced by a camera with pose C_{k-1} at time instant $k-1$. At the next time instant, k, the camera has pose C_k and produces image I_k. Poses C_{k-1} and C_k can be related by

$$C_k = C_{k-1}T_{k,k-1} \tag{2.1}$$

where $T_{k,k-1} \in \mathbb{R}^{4\times4}$ is a rigid body transformation of the following form:

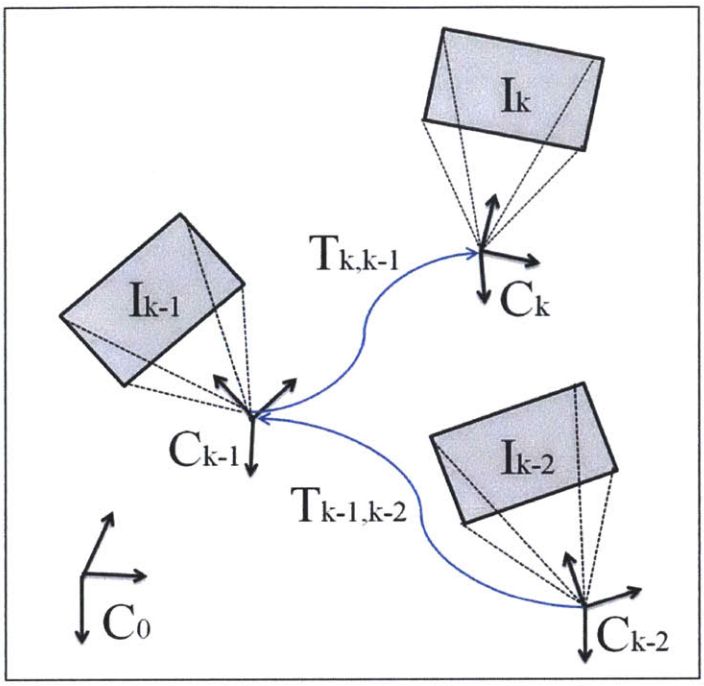


Figure 2-1: An illustration of the visual odometry problem. The relative transformations $T_{k,k-1}$ between adjacent camera poses C_{k-1} and C_k are computed from visual features in images I_{k-1} and I_k. All transformations are concatenated to get absolute pose C_k with respect to the initial coordinate frame C_0.

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

where $R_{k,k-1} \in SO(3)$ is the rotation matrix and $t_{k,k-1} \in \mathbb{R}^{3\times 1}$ the translation vector [43].

In visual odometry, the goal is to recover the entire camera trajectory by relating the set of all camera poses, $C_{0:n} = \{C_0, \ldots, C_n\}$, to the initial pose, C_0. For example, $C_1 = C_0 T_{1,0}$ and $C_2 = C_0 T_{1,0} T_{2,1}$, which can be rewritten as $C_2 = C_1 T_{2,1}$. Thus, the current pose, C_n, is computed by concatenating all the transformations $T_{k,k-1}$ where $k = 1 \ldots n$ [43], therefore:

$$C_n = C_{n-1} T_{n,n-1} \tag{2.3}$$

The block diagram in figure 2-2 summarizes the VO process. First, features are detected in every image I_k and then matched with those from previous frames. Next,

relative motion T_k between the time instants $k-1$ and k is computed. The camera pose C_k is then computed according to equation 2.1. Finally, local optimization over the last m frames can be performed to obtain a more accurate estimate of the local trajectory [43]. Each step will now be discussed briefly.

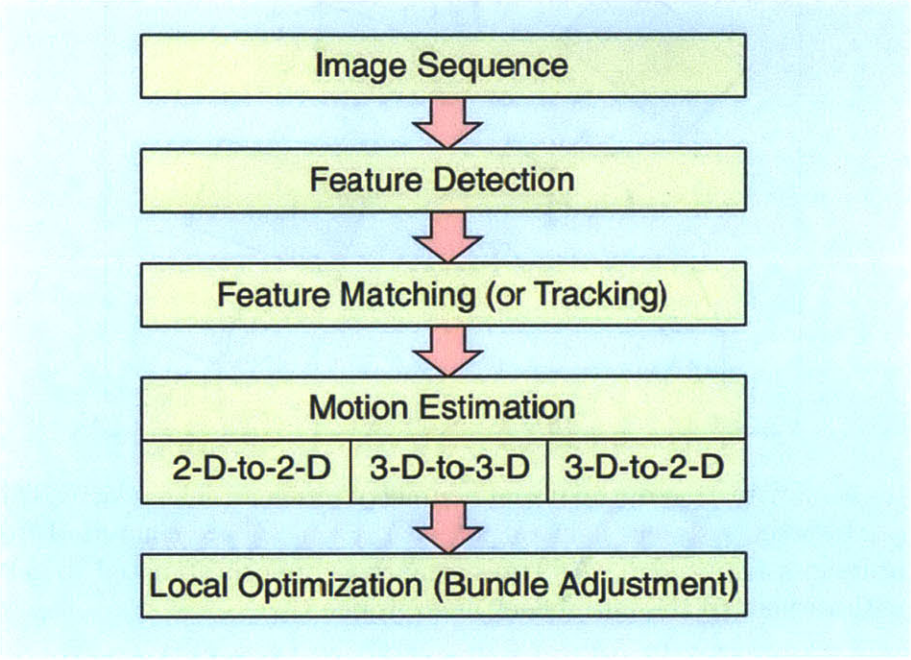


Figure 2-2: A block diagram showing the main components of a VO system [43].

Feature Detection

A feature is an image pattern that is unique to its immediate surrounding due to intensity, color and texture. In this first step of the VO process, a feature detector searches for all the point-features in an image. Point-features, such as blobs and corners (the intersection point of two or more edges), are useful in VO because they can be accurately located within an image. There are many different point-feature detectors; Harris, Shi-Tomasi, Moravec and FAST are examples of corner detectors [17, 44, 33, 41] while SIFT, SURF, and CENSURE are types of blob detectors [30, 4, 1]. The appropriate choice of feature detector depends on the environment and computational constraints. Although each detector has its own advantages and disadvantages, the desirable properties of a feature detector are [14]:

1. localization accuracy both in position and scale

2. repeatability (many features should be redetected in the next images)

3. computational efficiency

4. robustness to noise, compression artifacts, blur

5. distinctiveness (so features can be accurately matched across different images)

6. invariance to both photometric and geometric changes

Every feature detector has the same general procedure. First, a feature-response function is applied to the entire image. The type of function used is one element that differentiates the feature detectors (i.e. the Harris detector uses a corner response function while the SIFT detector uses the difference-of-Gaussian detector). Next, all the local minima or maxima of the feature-response function are identified. These points are the detected features. Finally, the region surrounding each feature is assigned a descriptor, e.g. pixel intensity, so that it can be matched to descriptors from other image frames [14].

Feature Matching or Feature tracking

There are many methods to match features between images; the simplest way is to compare all feature descriptors from one image to all feature descriptors from a second image using some kind of similarity measure (i.e. sum of squared differences or normalized cross correlation) [14]. The type of descriptor influences the choice of similarity measure. Another option for feature matching is to search for all features in one image and then search for those features in other images. This "detect-then-track" method is preferable when motion and change in appearance between frames is small [14]. The set of all matches corresponding to a single feature is called a feature track.

Motion Estimation

The goal of the motion estimation step is to use all of the feature tracks to compute the camera motion from the previous image frame to the current frame. Consider f_k, the set of features in the current frame I_k. The 3D location of the ith feature, f_k^i, is represented as 3D coordinate points, $\tilde{X}_k^i$. Similarly, the ith feature in frame I_{k-1} is represented as 3D coordinate points $\tilde{X}_{k-1}^i$. Camera motion is computed by finding the transformation from the set of 3D features from image I_k to the set of 3D features from image I_{k-1}. Explicitly, the solution is the $T_{k,k-1}$ that minimizes the L_2 distance between the two 3-D feature sets:

$$\hat{T}_{k,k-1} = \arg\min_{T_{k,k-1}} \sum_i ||\tilde{X}_k^i - T_{k,k-1}\tilde{X}_{k-1}^i|| \tag{2.4}$$

These transformations have absolute scale since the features are represented as 3D points [43]. Then, the current pose, C_n, is computed directly by concatenating all the transformations $T_{k,k-1}$ (where $k = 1 \ldots n$) as in equation 2.1.

There are two additional ways to represent features. In one case, both f_{k-1} and f_k are specified using 2-D image coordinates. Alternatively, f_{k-1} can be specified in 3-D coordinates and f_k as the corresponding 2-D reprojections on the image I_k. If either of these feature representations are used, slightly different methods for estimating motion are used. These methods are explained in [43].

Local Optimization (Bundle Adjustment)

The advantage of performing alignment only between the last pair of images is that the computational complexity of the algorithm has constant time with respect to path length. The disadvantage is that drift is introduced in the trajectory as small errors in position and rotation build up over time [11]. One way to alleviate this is to use windowed bundle adjustment, which optimizes the camera parameters and the 3-D feature parameters simultaneously, for a set of m image frames [14]. Specifically, windowed bundle adjustment aims to minimize the image reprojection error:

$$\left[\hat{X}, \hat{C}_{1:k}\right] = \arg\min_{X,C} \sum_{i=1}^{n} \sum_{k=1}^{m} ||p_k^i - g(X^i, C_k)||^2 \tag{2.5}$$

where n is the number of features, p_k^i is the ith image point of the 3-D feature X^i measured in the kth image and $g(X^i, C_k)$ is its image reprojection according to current camera pose C_k [14]. Since windowed bundle adjustment uses feature measurements over m image frames, drift is reduced compared to two-view VO. As m increases, drift is reduced, but computational expense increases. Thus, the choice of window size, m, is chosen based on computational requirements [14].

2.1.2 Visual-Inertial Odometry

Visual odometry alone has significant latency, computational complexity and potential robustness issues depending on the environment [19]. These characteristics make vision a natural supplement to inertial measurement units (IMUs) in aiding navigation [34, 36, 19]. Visual-inertial odometry (VIO), the process of estimating a vehicle's trajectory using both inertial measurements and feature observations from cameras, is an extension of the VO problem previously described.

Inertial Measurement Units

Inertial measurement units (IMUs) are comprised of accelerometers, which measure the acceleration of the IMU, and gyroscopes, which measure the angular velocity of the IMU. From these measurements, position, velocity and attitude of the vehicle can be calculated using integration. IMUs are self contained and are capable of providing measurements at high sample rates [25]. IMU measurements are subject to biases in both the accelerometers and gyroscopes as well as random walk produced by integration of the intrinsic noise present in the accelerometers and gyroscopes [38].

VIO methods

As discussed in section 1.2, there are several methods for combining visual and inertial measurements to perform visual-inertial odometry. Loosely coupled methods are

computationally efficient but are less accurate. Tightly coupled methods, further classified as either EKF-based or iterative minimization methods, are more accurate. The MSCKF is the basis for the VIO method presented in this thesis; thus, EKF-based methods, and the MSCKF in particular, will be described in detail in chapter 3.

2.2 Simultaneous Localization and Mapping (SLAM)

The goal of simultaneous localization and mapping (SLAM) is for a mobile robot in an unknown environment at an unknown location to incrementally build a consistent map of the surrounding environment while, simultaneously, localizing itself within the map. The SLAM problem originated in the late 1980's and then a landmark paper by Randal Smith et al. showed that estimates of landmarks taken by a robot in an unknown environments are correlated with each other because of the error in estimated vehicle location [46]. This implied that a solution to the SLAM problem requires a joint state, comprised of the robot pose and each landmark position, to be updated at every landmark observation [12].

The probabilistic formulation of the SLAM problem will be presented in section 2.2.1. In this problem, the current vehicle pose and the feature locations are estimated from all the previous measurements and control inputs. The EKF-SLAM algorithm will be discussed in section 2.2.2. This algorithm applies the EKF to the SLAM problem using maximum likelihood data association [48]. It should be noted that though EKF-SLAM is considered a solution to the SLAM problem at a theoretical and conceptual level, substantial issues still exist in practically realizing general SLAM solutions [12]. Real-time implementation of EKF-SLAM algorithms, for example, might not be possible since the computation time scales quadratically with the number of landmarks in the map [3, 48]. Additionally, the loop-closure problem (i.e. the robot recognizing when it has returned to a previously mapped region), is a major hurdle in solving the SLAM problem [3],[12].

Distinction from VO

Solving the SLAM problem requires obtaining a global, consistent estimate of the robot path. In this problem, it is necessary to store a map of the environment so the robot recognizes when it has returned to a previously visited location. Visual odometry (and visual-inertial odometry) on the other hand, recovers the robot trajectory incrementally, pose after pose, with the option of locally optimizing trajectory estimate over the last m poses. Essentially, VO or VIO can be a single step with the entire SLAM problem [43].

2.2.1 Probabilistic SLAM

The SLAM problem requires a robot to estimate landmark locations and its own position in a previously unknown environment simultaneously. To solve this estimation problem, the following parameters are defined below [12]. An illustration of the problem follows in figure 2-3.

List of Parameters

$\mathbf{x}_k$: State vector describing the location and orientation of the robot at time k

$\mathbf{u}_k$: Control vector, applied at time $k-1$ to drive the robot to state $\mathbf{x}_k$ at time k

$\mathbf{m}_i$: Vector describing the location (assumed to be time invariant) of the ith landmark

$\mathbf{z}_{ik}$: Observation of the ith landmark taken from the robot at time k

$\mathbf{z}_k$: Observation of multiple landmarks at time k

$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \ldots, \mathbf{x}_k\}$: History of all robot locations

$\mathbf{U}_{0:k} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$: History of control inputs

$\mathbf{m} = \{\mathbf{m}_1, \ldots, \mathbf{m}_n\}$: Set of all landmarks

$\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$: Set of all landmark observations

General SLAM Algorithm

An estimate for the distribution $p(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)$ at time $k-1$, the control input $\mathbf{u}_k$, and the observation $\mathbf{z}_k$ are used to compute the next probability distribution,

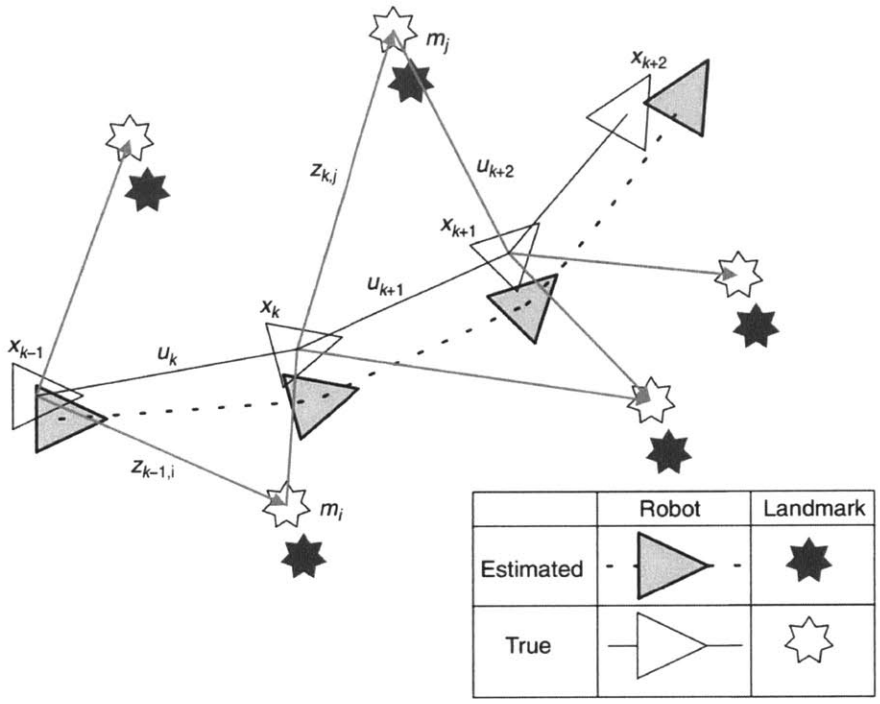


Figure 2-3: The essential SLAM problem. A simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations[12].

$p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$. This describes the joint posterior density of the landmark locations and vehicle state at time k, given the recorded observations, all control inputs and the initial state of the vehicle [12].

The SLAM algorithm is a standard two-step procedure consisting of a propagation step and a measurement update. First, in the propagation step, the distribution of $(\mathbf{x}_k, \mathbf{m})$ given all prior observations $(\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ is calculated. From Bayesian statistics, the marginal posterior predictive distribution is:

$$p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) p(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \tag{2.6}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ is the conditional probability distribution of the vehicle state, $\mathbf{x}_k$, given the previous state $\mathbf{x}_{k-1}$ and the control input $\mathbf{u}_k$.

The second step of the SLAM algorithm is the measurement update step. In this

step, Bayes theorem[1] is used to calculate the probability distribution:

$$p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{2.7}$$

where $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$ is the conditional probability distribution of a landmark measurement $\mathbf{z}_k$ given the vehicle location, $\mathbf{x}_k$, and landmark locations, $\mathbf{m}$.

The probability distribution $p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ is computed at all times k by repeating this two step recursive procedure; thereby yielding the solution to the SLAM problem.

2.2.2 EKF-SLAM

The EKF-SLAM algorithm applies the EKF to the SLAM problem. The inherent nature of the EKF algorithm requires a Gaussian noise assumption for vehicle motion and observation [48]. The nonlinear SLAM problem (i.e. vehicle motion and observations are nonlinear models) is also linearized by applying the EKF. Despite these assumptions and approximations, the EKF-SLAM algorithm has been successfully used in many robotic mapping problems [48].

EKF-SLAM Problem Formulation

The goal of the EKF-SLAM problem is to find the best estimates for the current vehicle state, $\hat{\mathbf{x}}_k$, and the landmark locations, $\hat{\mathbf{m}}_{1:N}$, using all the measurements, $\mathbf{Z}_{0:k}$. Specifically, the EKF-SLAM problem requires solving the optimization problem:

$$[\hat{\mathbf{x}}_k, \hat{\mathbf{m}}_{1:N}] = \arg\max_{\mathbf{x}_k, \mathbf{m}} p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}) \tag{2.8}$$

By applying Bayes theorem to the optimization problem 2.8, the probabilistic SLAM problem is recovered.

$$[\hat{\mathbf{x}}_k, \hat{\mathbf{m}}_{1:N}] = \arg\max_{\mathbf{x}_k, \mathbf{m}} \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \tag{2.9}$$

[1] Bayes theorem states that for events A and B, with $P(B) \neq 0$, then $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

As in the probabilistic SLAM formulation, the propagation is written as:

$$p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)p(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1})d\mathbf{x}_{k-1} \tag{2.10}$$

Now, the Gaussian assumptions for vehicle motion and observations are applied. Thus, by applying the Gaussian assumption $\mathbf{x}_{k-1} \sim N(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1})$ to $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ and $p(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1})$, it is true that $\mathbf{x}_k$ is also Gaussian with $\mathbf{x}_k \sim N(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$ [48]. Similarly, in the measurement update equation the Gaussian assumption $\mathbf{z}_k \sim N(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$ is applied to $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$

Thus, $\mathbf{x}_k$ and $\mathbf{z}_k$ can be written in the form [12]:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{2.11}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k \tag{2.12}$$

where $\mathbf{f}(\cdot)$ models vehicle kinematics and where $\mathbf{w}_k$ are additive, zero mean uncorrelated Gaussian motion disturbances with covariance $\mathbf{Q}_k$. Additionally, $\mathbf{h}(\cdot)$ describes the geometry of the observation and $\mathbf{v}_k$ are additive, zero mean uncorrelated Gaussian observation errors with covariance $\mathbf{R}_k$.

EKF-SLAM Algorithm

With the SLAM problem now formulated as equations 2.11 and 2.12, the standard EKF method [15] can be applied to determine the mean, $\begin{bmatrix}\hat{\mathbf{x}}_{k|k} & \hat{\mathbf{m}}_k\end{bmatrix}^T$, and covariance, $\mathbf{P}_{k|k}$, of the joint posterior distribution $p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ at all times k.

$$\begin{bmatrix}\hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k\end{bmatrix} = E\begin{bmatrix}\mathbf{x}_k|\mathbf{Z}_{0:k} \\ \mathbf{m}|\mathbf{Z}_{0:k}\end{bmatrix} \tag{2.13}$$

$$\mathbf{P}_{k|k} = \begin{bmatrix}\mathbf{P}_{xx} & \mathbf{P}_{xm} \\ \mathbf{P}_{mx} & \mathbf{P}_{mm}\end{bmatrix}_{k|k} = E\left[\begin{pmatrix}\mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k\end{pmatrix}\begin{pmatrix}\mathbf{x}_k - \hat{\mathbf{x}}_k \\ \mathbf{m} - \hat{\mathbf{m}}_k\end{pmatrix}^T|\mathbf{Z}_{0:k}\right] \tag{2.14}$$

The following equations are the typical propagation equations for a discrete-time extended Kalman Filter:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \tag{2.15}$$

$$\mathbf{P}_{xx,k|k-1} = \nabla\mathbf{f}\mathbf{P}_{xx,k-1|k-1}\nabla\mathbf{f}^T + \mathbf{Q}_k \tag{2.16}$$

where $\nabla\mathbf{f}$ is the Jacobian of $\mathbf{f}$ evaluated at the estimate $\hat{\mathbf{x}}_{k-1|k-1}$. It is also noted that during propagation, the estimate of (stationary) landmark locations does not change [12].

Finally, the measurement-update equations are given as:

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{m}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{m}}_{k-1} \end{bmatrix} + \mathbf{K}_k\mathbf{r}_k \tag{2.17}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{2.18}$$

where

$$\begin{aligned} \mathbf{r}_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \hat{\mathbf{m}}_{k-1}) \\ \mathbf{S}_k &= \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \end{aligned} \tag{2.19}$$

In the measurement-update equations above, the measurement residual is $\mathbf{r}_k$ and $\mathbf{H}$ is the Jacobian of $\mathbf{h}$ evaluated at $\hat{\mathbf{x}}_{k|k-1}$ and $\hat{\mathbf{m}}_{k-1}$.

2.3 Summary

This chapter provided an overview of the VO problem (section 2.1) and the SLAM problem (section 2.2). The key components of the VO problem, including feature detection, feature matching and tracking, motion estimation and bundle adjustment, have all been briefly summarized. The probabilistic SLAM problem has been described and the general EKF-SLAM algorithm has been presented. The VO problem and the EKF-SLAM algorithm provide a framework for EKF-based VIO estimation, which is discussed in the following chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

EKF-Based VIO Estimation

The goal of visual inertial odometry is to estimate a moving vehicle's trajectory using inertial measurements and observations, obtained by a camera, of naturally occurring point features. In order to accomplish this in real-time, the VIO algorithm must have bounded computational complexity as a function of both time and trajectory length [27, 29]. EKF-based VIO algorithms, tightly coupled estimators with bounded computational cost, are generally divided into two categories: EKF-SLAM and sliding window. In EKF-SLAM algorithms, the state vector contains the vehicle state and the feature locations. In order to keep the computational cost bounded, features must be removed from the state vector once they leave the field of view of the camera [27]. In sliding window algorithms, the state vector maintains a sliding window of camera poses and uses feature observations to apply probabilistic constraints between the poses [29].

EKF-SLAM algorithms will be discussed in section 3.1. The multi-state constraint Kalman filter (MSCKF), a sliding window algorithm, is described in section 3.2. As a result of two key properties of the MSCKF, correct observability and consistency, the MSCKF outperforms other EKF-based algorithms in terms of accuracy and computational complexity. This is discussed when the two types of algorithms are compared in section 3.3.

3.1 EKF-SLAM Algorithms

In EKF-SLAM algorithms, the filter state is comprised of the IMU state and the feature locations. In these algorithms, features that move out of the camera's field of view are removed from the filter state vector in order to keep the computational cost of the algorithm bounded [29]. As in typical EKF algorithms, the state estimates are calculated using a two-step algorithm. First, is a state propagation step, followed by a measurement update step. The standard EKF-SLAM algorithm given in [29] is described in this section.

3.1.1 IMU State

The IMU state is defined with respect to a global reference frame $\{G\}$. The coordinate frame $\{I\}$ is affixed to the IMU and the rotation from the global frame to the IMU frame at time step ℓ is described by the unit quaternion ${}^{I_\ell}_{G}\bar{\mathbf{q}} \in \mathbb{R}^4$. The position and velocity of the IMU in the global frame are $\mathbf{p}_\ell \in \mathbb{R}^3$ and $\mathbf{v}_\ell \in \mathbb{R}^3$, respectively. Additionally, the biases of the gyroscope and accelerometer are $\mathbf{b}_{\mathbf{g}_\ell}$ and $\mathbf{b}_{\mathbf{a}_\ell}$, both in $\mathbb{R}^3$. These quantities form the 16×1 IMU state vector at time step ℓ:

$$\mathbf{x}_{I_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^T & \mathbf{p}_\ell^T & \mathbf{v}_\ell^T & \mathbf{b}_{\mathbf{g}_\ell}^T & \mathbf{b}_{\mathbf{a}_\ell}^T \end{bmatrix}^T \tag{3.1}$$

To define the IMU error state, additive error for position (${}^G\tilde{\mathbf{p}} = {}^G\mathbf{p} - {}^G\hat{\mathbf{p}}$), velocity and the biases is used. The orientation error is given by the minimal representation ${}^G\tilde{\boldsymbol{\theta}}^T \in \mathbb{R}^3$ [29]. Thus, the IMU error state is the 15×1 vector:

$$\tilde{\mathbf{x}}_I = \begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}^T & {}^G\tilde{\mathbf{p}}^T & {}^G\tilde{\mathbf{v}}^T & \tilde{\mathbf{b}}_{\mathbf{g}}^T & \tilde{\mathbf{b}}_{\mathbf{a}}^T \end{bmatrix}^T \tag{3.2}$$

3.1.2 EKF-SLAM Filter State

In EKF-SLAM algorithms, the filter state contains the current IMU state, $\mathbf{x}_{I_\ell}$, and the feature locations. The feature locations can be represented in different ways. For example, the location of ith feature, $\mathbf{f}_i$, can be represented using traditional 3-D

Euclidean XYZ parametrization or by the 6-DOF inverse depth parametrization [8]. The choice of parametrization can affect the filter's accuracy [29] and this will be discussed in more detail in section 3.1.4. Regardless of parametrization choice, each feature, $\mathbf{f}_i$ for $i = 1, \ldots, n$ is included in the state vector at time step ℓ. Therefore, the complete filter state vector at time-step ℓ is defined as:

$$\mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^T & \mathbf{f}_1^T & \ldots & \mathbf{f}_n^T \end{bmatrix}^T \tag{3.3}$$

3.1.3 Discrete-Time Propagation

In the propagation step of EKF-SLAM algorithms, the inertial measurements from the IMU's gyroscope and accelerometer are used to propagate the filter state. The procedure for the state propagation is presented in this section.

System Model

The system consists of the inertial measurement signals and the IMU state evolution. In continuous time, the gyroscope measurements, $\boldsymbol{\omega}_m$, and the accelerometer measurements, $\mathbf{a}_m$, are given by [6]:

$$\begin{aligned} \boldsymbol{\omega}_m &= {}^I\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\ \mathbf{a}_m &= {}^I_G\mathbf{R}({}^G\mathbf{a} - {}^G\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \end{aligned} \tag{3.4}$$

where $\mathbf{n}_g$ and $\mathbf{n}_a$ are zero-mean white Gaussian noise processes modeling measurement noise. Additionally, ${}^I_G\mathbf{R} = {}^I_G\mathbf{R}({}^{I_\ell}_G\bar{\mathbf{q}})$ is a rotational matrix from the global to IMU frame, ${}^G\mathbf{g}$ is the gravity vector in the global frame, ${}^G\mathbf{a}$ is the body acceleration in the global frame, ${}^I\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ is the rotational velocity in the IMU frame and

$$\lfloor {}^I\boldsymbol{\omega} \times \rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.5}$$

The IMU state evolution is described by the equations [6]:

$$
\begin{aligned}
{}^{I}_{G}\dot{\bar{\mathbf{q}}}(t) &= \frac{1}{2}\mathbf{\Omega}({}^{I}\boldsymbol{\omega}(t)){}^{I}_{G}\bar{\mathbf{q}}(t) \\
{}^{G}\dot{\mathbf{p}}_{I}(t) &= {}^{G}\mathbf{v}_{I}(t) \\
{}^{G}\dot{\mathbf{v}}_{I}(t) &= {}^{G}\mathbf{a}_{I}(t) \\
\dot{\mathbf{b}}_{g}(t) &= \mathbf{n}_{g}(t) \\
\dot{\mathbf{b}}_{a}(t) &= \mathbf{n}_{a}(t)
\end{aligned}
\tag{3.6}
$$

where $\mathbf{\Omega}({}^{I}\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor {}^{I}\boldsymbol{\omega} \times \rfloor & {}^{I}\boldsymbol{\omega} \\ -{}^{I}\boldsymbol{\omega}^{T} & 0 \end{bmatrix}$.

Note that the features are assumed to be stationary, so the feature location estimates are not propagated from the inertial measurements.

IMU State Estimate Propagation

The continuous time equations for propagating estimates of the IMU state can easily be derived by applying the expectation operator to the state propagation equations 3.6; however, in practice, the signals $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$ are only sampled at discrete times $t_{\ell-1}$ and t_ℓ. Therefore, the signal measurements from time $t_{\ell-1}$ are used to propagate the IMU state estimate, $\hat{\mathbf{x}}_{I_{\ell-1|\ell-1}}$, from time $t_{\ell-1}$ to time t_ℓ, thereby producing the state estimate $\hat{\mathbf{x}}_{I_{\ell|\ell-1}}$. The discrete-time equations for propagating the estimates of the time-evolving IMU state are given by [29]:

$$
\begin{aligned}
{}^{I_\ell}_{G}\hat{\bar{\mathbf{q}}} &= {}^{I_\ell}_{I_{\ell-1}}\hat{\bar{\mathbf{q}}} \otimes {}^{I_{\ell-1}}_{G}\hat{\bar{\mathbf{q}}} \\
{}^{G}\hat{\mathbf{p}}_{\ell} &= {}^{G}\hat{\mathbf{p}}_{\ell-1} + {}^{G}\hat{\mathbf{v}}_{\ell-1}\Delta t + {}^{G}_{I_{\ell-1}}\hat{\mathbf{R}}\hat{\mathbf{y}}_{\ell-1} + \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2 \\
{}^{G}\hat{\mathbf{v}}_{\ell} &= {}^{G}\hat{\mathbf{v}}_{\ell-1} + {}^{G}_{I_{\ell-1}}\hat{\mathbf{R}}\hat{\mathbf{s}}_{\ell-1} + {}^{G}\mathbf{g}\Delta t \\
\hat{\mathbf{b}}_{\mathbf{g}_\ell} &= \hat{\mathbf{b}}_{\mathbf{g}_{\ell-1}} \\
\hat{\mathbf{b}}_{\mathbf{a}_\ell} &= \hat{\mathbf{b}}_{\mathbf{a}_{\ell-1}}
\end{aligned}
\tag{3.7}
$$

where ${}^{I_\ell}_{I_{\ell-1}}\hat{\bar{\mathbf{q}}}$ solves the differential equation ${}^{I_t}_{I_{\ell-1}}\dot{\hat{\bar{\mathbf{q}}}} = \frac{1}{2}\mathbf{\Omega}(\hat{\boldsymbol{\omega}}(t)){}^{I_t}_{I_{\ell-1}}\hat{\bar{\mathbf{q}}}$ with $t \in [t_{\ell-1}, t_\ell]$ and initial condition ${}^{I_{\ell-1}}_{I_{\ell-1}}\hat{\bar{\mathbf{q}}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ with $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}_g}$. Also, $\Delta t = t_\ell - t_{\ell-1}$, ${}^{G}_{I_{\ell-1}}\hat{\mathbf{R}} = {}^{G}_{I}\hat{\mathbf{R}}(t_{\ell-1})$, and

$$\hat{\mathbf{s}}_{\ell-1} = \int_{t_{\ell-1}}^{t_\ell} {}^{I_{\ell-1}}_{I_\tau}\hat{\mathbf{R}}(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_a(\tau))d\tau$$
$$\hat{\mathbf{y}}_{\ell-1} = \int_{t_{\ell-1}}^{t_\ell}\int_{t_{\ell-1}}^{s} {}^{I_{\ell-1}}_{I_\tau}\hat{\mathbf{R}}(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_a(\tau))d\tau ds \tag{3.8}$$

A complete derivation of equations 3.7 can be found in [26].

IMU Error State Propagation

The IMU state error propagation depends on errors at the previous step. A linearized equation of the following form is used in the EKF [29]:

$$\tilde{\mathbf{x}}_{I_{\ell|\ell-1}} \simeq \mathbf{\Phi}_{I_{\ell-1}}\tilde{\mathbf{x}}_{I_{\ell-1|\ell-1}} + \mathbf{w}_{d_{\ell-1}} \tag{3.9}$$

where $\mathbf{\Phi}_{I_{\ell-1}}$ is the IMU error-state transition matrix and $\mathbf{w}_{d_{\ell-l}}$ is a noise vector with covariance matrix $\mathbf{Q}_{d_{\ell-1}}$. $\mathbf{\Phi}_{I_{\ell-1}}$ is computed by linearizing the state propagation equations 3.7 and is given by [29]:

$$\mathbf{\Phi}_{I_{\ell-1}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{qb_g} & \mathbf{0}_3 \\ \mathbf{\Phi}_{pq} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{\Phi}_{pb_g} & \mathbf{\Phi}_{pa} \\ \mathbf{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{vb_g} & \mathbf{\Phi}_{va} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{3.10}$$

where

$$
\begin{aligned}
\mathbf{\Phi}_{pq} &= -\lfloor ({}^{G}\hat{\mathbf{p}}_{\ell} - {}^{G}\hat{\mathbf{p}}_{\ell-1} - {}^{G}\hat{\mathbf{v}}_{\ell-1}\Delta t - \frac{1}{2}\mathbf{g}\Delta t^2) \times \rfloor \\
\mathbf{\Phi}_{vq} &= -\lfloor ({}^{G}\hat{\mathbf{v}}_{\ell} - {}^{G}\hat{\mathbf{v}}_{\ell-1} - \mathbf{g}\Delta t) \times \rfloor \\
\mathbf{\Phi}_{gb_g} &= {}^{I_{\ell-1}}_{G}\hat{\mathbf{R}}^T \int_{t_{\ell-1}}^{t_{\ell}} {}^{I_{\ell-1}}_{I_\tau}\hat{\mathbf{R}} d\tau \\
\mathbf{\Phi}_{pb_g} &= \int_{t_{\ell-1}}^{t_{\ell}} \int_{t_{\ell-1}}^{w} \lfloor ({}^{G}\dot{\hat{\mathbf{v}}}_{\tau} - \mathbf{g}) \times \rfloor {}^{I_{\ell-1}}_{G}\hat{\mathbf{R}}^T \int_{t_{\ell-1}}^{\tau} {}^{I_{\ell-1}}_{I_s}\hat{\mathbf{R}} ds d\tau dw \\
\mathbf{\Phi}_{pa} &= {}^{I_{\ell-1}}_{G}\hat{\mathbf{R}}^T \int_{t_{\ell-1}}^{t_{\ell}} \int_{t_{\ell-1}}^{\tau} {}^{I_{\ell-1}}_{I_s}\hat{\mathbf{R}} ds d\tau \\
\mathbf{\Phi}_{vb_g} &= \int_{t_{\ell-1}}^{t_{\ell}} \lfloor ({}^{G}\dot{\hat{\mathbf{v}}}_{\tau} - \mathbf{g}) \times \rfloor {}^{I_{\ell-1}}_{G}\hat{\mathbf{R}}^T \int_{t_{\ell-1}}^{\tau} {}^{I_{\ell-1}}_{I_s}\hat{\mathbf{R}} ds d\tau \\
\mathbf{\Phi}_{va} &= {}^{I_{\ell-1}}_{G}\hat{\mathbf{R}}^T \int_{t_{\ell-1}}^{t_{\ell}} {}^{I_{\ell-1}}_{I_\tau}\hat{\mathbf{R}} d\tau
\end{aligned}
\tag{3.11}
$$

A complete derivation of equations 3.10 and 3.11 can be found in [26].

Filter Covariance Propagation

The filter covariance matrix is propagated as [29]:

$$
\mathbf{P}_{\ell|\ell-1} = \begin{bmatrix} \mathbf{\Phi}_{I_{\ell-1}} \mathbf{P}_{II_{\ell-1|\ell-1}} \mathbf{\Phi}_{I_{\ell-1}}{}^{T} + \mathbf{Q}_{d_{\ell-1}} & \mathbf{\Phi}_{I_{\ell-1}} \mathbf{P}_{IF_{\ell-1|\ell-1}} \\ \mathbf{P}_{IF_{\ell-1|\ell-1}}{}^{T} \mathbf{\Phi}_{I_{\ell-1}}{}^{T} & \mathbf{P}_{FF_{\ell-1|\ell-1}} \end{bmatrix}
\tag{3.12}
$$

where $\mathbf{P}_{II_{\ell-1|\ell-1}}$ is the covariance matrix of the IMU state, $\mathbf{P}_{FF_{\ell-1|\ell-1}}$ is the covariance of the features and $\mathbf{P}_{IF_{\ell-1|\ell-1}}$ the cross-covariance between them.

3.1.4 Observation Update

In EKF-SLAM algorithms, observation updates occur whenever a new image is obtained from the camera. The location of each feature in the image is determined and compared to the corresponding expected location in the image. Since the measurement function is nonlinear, a linearized approximation is used to obtain a linearized residual. The set of residuals is then used in a standard EKF update. The observation update procedure is now presented.

Feature Residual Calculation

The observation of a feature i at time step ℓ is:

$$\mathbf{z}_{i_l} = \mathbf{h}(\mathbf{x}_{I_\ell}, \mathbf{f}_i) + \mathbf{n}_{i\ell} \tag{3.13}$$

where $\mathbf{n}_{i\ell}$ is the measurement noise vector, modeled as zero-mean Gaussian with covariance matrix $\mathbf{R}_{i\ell} = \sigma^2\mathbf{I}_2$.

The measurement function $\mathbf{h}$ depends on the camera system. For example, in monocular systems, the measurement of a feature is two-dimensional and the measurement model is [29]:

$$\mathbf{z}_{i_l} = \begin{bmatrix} \frac{{}^{C_\ell}x_{f_i}}{{}^{C_\ell}z_{f_i}} \\ \frac{{}^{C_\ell}y_{f_i}}{{}^{C_\ell}z_{f_i}} \end{bmatrix} + \mathbf{n}_{i\ell} \tag{3.14}$$

where the vector ${}^{C_\ell}\mathbf{p}_{f_i} = \begin{bmatrix} {}^{C_\ell}x_{f_i} & {}^{C_\ell}y_{f_i} & {}^{C_\ell}z_{f_i} \end{bmatrix}^T$ is the position of the feature with respect to the camera at time step ℓ.

The residual between the actual and expected feature measurement is:

$$\mathbf{r}_{i\ell} = \mathbf{z}_{i\ell} - \mathbf{h}(\hat{\mathbf{x}}_{I_{\ell|\ell-1}}, \hat{\mathbf{f}}_{i_{\ell|\ell-1}}) \tag{3.15}$$

and the linearized approximation is:

$$\mathbf{r}_{i\ell} \simeq \mathbf{H}_{i\ell}(\hat{\mathbf{x}}_{\ell|\ell-1})\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{n}_{i\ell} \tag{3.16}$$

where $\mathbf{H}_{i\ell}(\hat{\mathbf{x}}_{\ell|\ell-1})$ is the Jacobian matrix of $\mathbf{h}$ with respect to the filter state, evaluated at the state estimate $\hat{\mathbf{x}}_{\ell|\ell-1}$. Note that $\hat{\mathbf{x}}_{\ell|\ell-1}$ and $\tilde{\mathbf{x}}_{\ell|\ell-1}$ are determined from equations 3.7 and 3.9, respectively, in the discrete-time propagation step.

EKF Update

The residual corresponding to each feature undergoes a Mahalanobis gating test [29] in order to remove any outliers from being used in the EKF update. If the feature

passes the test, it is used in the EKF update. The following matricies are formed:

$$\mathbf{r}_\ell = \begin{bmatrix} \mathbf{r}_{1\ell} & \dots & \mathbf{r}_{i\ell} & \dots & \mathbf{r}_{k\ell} \end{bmatrix}^T \tag{3.17}$$

$$\mathbf{H}_\ell = \begin{bmatrix} \mathbf{H}_{1\ell} & \dots & \mathbf{H}_{i\ell} & \dots & \mathbf{H}_{k\ell} \end{bmatrix}^T \tag{3.18}$$

$$\mathbf{R}_\ell = \begin{bmatrix} \mathbf{R}_{1\ell} & & 0 \\ & \ddots & \\ 0 & & \mathbf{R}_{k\ell} \end{bmatrix} \tag{3.19}$$

where $1 \leq k \leq n$ are the accepted features.

Finally, $\mathbf{r}_\ell$, $\mathbf{H}_\ell$ and $\mathbf{R}_\ell$ are used to perform a standard EKF observation update [15] as described in section 2.2.2:

$$\hat{\mathbf{x}}_{\ell|\ell} = \hat{\mathbf{x}}_{\ell|\ell-1} + \mathbf{K}_\ell \mathbf{r}_\ell \tag{3.20}$$

$$\mathbf{P}_{\ell|\ell} = \mathbf{P}_{\ell|\ell-1} - \mathbf{K}_\ell \mathbf{S}_\ell \mathbf{K}_\ell^T \tag{3.21}$$

where

$$\mathbf{S}_\ell = \mathbf{H}_\ell \mathbf{P}_{\ell|\ell-1} \mathbf{H}_\ell^T + \mathbf{R}_\ell \tag{3.22}$$

$$\mathbf{K}_\ell = \mathbf{P}_{\ell|\ell-1} \mathbf{H}_\ell^T \mathbf{S}_\ell^{-1} \tag{3.23}$$

3.2 MSCKF

The multi-state constraint Kalman filter, presented by Mourikis [34, 29], is an EKF-based VIO algorithm that differs from the standard EKF-SLAM algorithms in the way feature measurements are used. Like EKF-SLAM algorithms, the MSCKF algorithm consists of a propagation step followed by a measurement update step; the difference is that in the MSCKF algorithm, the state vector maintains a sliding window of camera poses and uses feature observations to apply probabilistic constraints between the poses rather than keeping feature locations in the state vector [29]. The MSCFK, as

presented in [29], is described in this section.

3.2.1 IMU state Augmentation

The IMU state of the MSCKF is the same IMU state used in EKF-SLAM algorithms (given in section 3.1.1) augmented with the pose of the camera with respect to the IMU.

$$\boldsymbol{\pi}_{CI} = \left\{ {}^{C}_{I}\bar{\mathbf{q}}, \;\; {}^{C}\mathbf{p}_{I} \right\} \tag{3.24}$$

where ${}^{C}_{I}\bar{\mathbf{q}} \in \mathbb{R}^4$ the unit quaternion describing the rotation from the IMU frame to the camera frame and ${}^{C}\mathbf{p}_{I} \in \mathbb{R}^3$ is the position of the IMU in the camera frame. By including the camera-to-IMU transformation in the state vector, the transformation can be estimated in unknown environments, with no *a priori* known features; something EKF-SLAM algorithms are unable to do [29]. The IMU state in the MSCKF is the 23×1 vector:

$$\mathbf{x}_{I_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_{\ell}^T & {}^{G}\mathbf{v}_{\ell}^T & \mathbf{b}_{\mathbf{g}_\ell}^T & \mathbf{b}_{\mathbf{a}_\ell}^T & {}^{C}_{I}\bar{\mathbf{q}}^T & {}^{C}\mathbf{p}_{I}^T \end{bmatrix}^T \tag{3.25}$$

3.2.2 MSCKF Filter State

The key difference between the MSCKF and EKF-SLAM algorithms is that the MSCKF state vector contains the IMU poses at the times the last N images were recorded. Thus, the state vector at time step ℓ is:

$$\mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^T & \boldsymbol{\pi}_{\ell-1}^T & \boldsymbol{\pi}_{\ell-2}^T & \cdots & \boldsymbol{\pi}_{\ell-N}^T \end{bmatrix}^T \tag{3.26}$$

where $\boldsymbol{\pi}_j = \begin{bmatrix} {}^{I_j}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_{j}^T \end{bmatrix}^T$, for $j = \ell - N, \ldots, \ell - 1$, are the recorded IMU poses at the times the last N images were recorded.

3.2.3 Discrete-Time Propagation

The propagation step of the MSCFK algorithm is exactly the same as in EKF-SLAM algorithms [29]. The inertial measurements from the IMU's gyroscope and accelerom-

eter are used to propagate the filter state. Identical to EKF-SLAM algorithms, the IMU state, $\mathbf{x}_{I_{\ell-1}}^T$, is the only part of the filter state that is propagated (i.e. the IMU poses, $\boldsymbol{\pi}_j$, remain the same). It is also important to note that the estimate of the camera to IMU transformation, $\boldsymbol{\pi}_{CI}$, (which is included in the IMU state) does not change during propagation. Thus the state transition matrix is:

$$\boldsymbol{\Phi}_{I_{\ell-1}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\Phi}_{qb_g} & \mathbf{0}_3 & \mathbf{0}_{3\times6} \\ \boldsymbol{\Phi}_{pq} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \boldsymbol{\Phi}_{pb_g} & \boldsymbol{\Phi}_{pa} & \mathbf{0}_{3\times6} \\ \boldsymbol{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 & \boldsymbol{\Phi}_{vb_g} & \boldsymbol{\Phi}_{va} & \mathbf{0}_{3\times6} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times6} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3\times6} \\ \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{I}_6 \end{bmatrix} \tag{3.27}$$

Refer to section 3.1.3 and equation 3.11 for the definitions of the terms in $\boldsymbol{\Phi}_{I_{\ell-1}}$.

The filter covariance matrix is propagated as:

$$\mathbf{P}_{\ell|\ell-1} = \begin{bmatrix} \boldsymbol{\Phi}_{I_{\ell-1}} \mathbf{P}_{II_{\ell-1|\ell-1}} \boldsymbol{\Phi}_{I_{\ell-1}}{}^T + \mathbf{Q}_{d_{\ell-1}} & \boldsymbol{\Phi}_{I_{\ell-1}} \mathbf{P}_{I\pi_{\ell-1|\ell-1}} \\ \mathbf{P}_{I\pi_{\ell-1|\ell-1}}{}^T \boldsymbol{\Phi}_{I_{\ell-1}}{}^T & \mathbf{P}_{\pi\pi_{\ell-1|\ell-1}} \end{bmatrix} \tag{3.28}$$

where $\mathbf{P}_{II_{\ell-1|\ell-1}}$ is the covariance matrix of the IMU state, $\mathbf{P}_{\pi\pi_{\ell-1|\ell-1}}$ is the covariance of the IMU poses and $\mathbf{P}_{I\pi_{\ell-1|\ell-1}}$ the cross-covariance between them. Note that this is the same covariance propagation as the EKF-SLAM algorithm in section 3.1.3; but the covariance matrix is of the IMU state and the IMU poses rather than the IMU state and feature locations.

3.2.4 Observation Update

In the MSCKF algorithm, observation updates occur when a feature is no longer detected, either because it has left the field of view of the camera, was occluded, or because the detector failed to find it. First, the location of the feature is estimated using all the feature's measurements via Gauss Newton minimization [29]. The measurement residuals are then computed from the estimated feature position. Since the measurement function is nonlinear, a linearized approximation is used to obtain a

linearized residual. Next, linearized constraints between all the camera poses from which the feature was observed are formed from the residuals. Finally, these constraints, which are independent of the errors in the feature coordinates, are used for the EKF update. The observation update procedure is now presented.

Estimation of Feature Location

To present the update algorithm, consider feature f_i, observed from the N poses in the MSCKF state vector, used for an update at time step ℓ. Note that the first observation of f_i occurs at time step $\ell - N$, and the Nth observation of f_i occurs at time step $\ell - 1$. Therefore, observations of f_i occur at times j, where $j = \ell - N, \ldots, \ell - 1$.

The first step is to obtain an estimate of the feature position ${}^G\hat{\mathbf{p}}_{f_i}$. In monocular systems, the measurement of a feature is two-dimensional and the jth observation of feature f_i observed from camera pose C_j is:

$$\mathbf{z}_{ij} = \mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i) + \mathbf{n}_{ij} = \frac{1}{{}^{C_j}z_{f_i}} \begin{bmatrix} {}^{C_j}x_{f_i} \\ {}^{C_j}y_{f_i} \end{bmatrix} + \mathbf{n}_{ij} \tag{3.29}$$

where $\mathbf{n}_{ij}$ is the measurement noise vector, modeled as zero-mean Gaussian with covariance matrix $\mathbf{R}_{ij} = \sigma^2\mathbf{I}_2$.

The position of the feature in the camera frame is:

$${}^{C_j}\mathbf{p}_{f_i} = \begin{bmatrix} {}^{C_j}x_{f_i} \\ {}^{C_j}y_{f_i} \\ {}^{C_j}z_{f_i} \end{bmatrix} = {}^C_I\mathbf{R}_{j|\ell-1}\,{}^I_G\mathbf{R}_{j|\ell-1}({}^G\mathbf{p}_{f_i} - {}^G\mathbf{p}_{j|\ell-1}) + {}^C\mathbf{p}_I \tag{3.30}$$

where ${}^G\mathbf{p}_{f_i}$ is the position of the feature in the global frame and is unknown. Therefore, an estimate ${}^G\hat{\mathbf{p}}_{f_i}$ is obtained using the measurements, $\mathbf{z}_{ij}$ and the filter estimates of the camera poses. This can be accomplished using Gauss-Newton minimization [29] or least-squares minimization [34].

Feature Residual Calculation

This section summarizes the procedure, given in [29], for calculating feature residuals. The residual (for $j = \ell - N, \ldots, \ell - 1$) between the actual and expected feature measurement is:

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, \hat{\boldsymbol{\pi}}_{CI_{j|\ell-1}}, {}^{G}\hat{\mathbf{p}}_{f_i}) \tag{3.31}$$

The linearized approximation is

$$\mathbf{r}_{ij} \simeq \mathbf{H}_{\boldsymbol{\pi}_{ij}} \tilde{\boldsymbol{\pi}}_{j|\ell-1} + \mathbf{H}_{C_{ij}} \tilde{\boldsymbol{\pi}}_{CI_{j|\ell-1}} + \mathbf{H}_{\mathbf{f}_{ij}} {}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{ij} \tag{3.32}$$

where $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and ${}^{G}\tilde{\mathbf{p}}_{f_i}$ are the errors of the current estimate for the jth pose and the error in the feature position respectively. $\tilde{\boldsymbol{\pi}}_{CI_{j|\ell-1}}$ is the camera-to-IMU transformation error.

$\mathbf{H}_{\boldsymbol{\pi}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ are the corresponding Jacobians evaluated using $\hat{\boldsymbol{\pi}}_{j|\ell-1}$ and ${}^{G}\hat{\mathbf{p}}_{f_i}$. Additionally, $\mathbf{H}_{C_{ij}}$ is the Jacobian of the measurement with respect to the camera-IMU pose evaluated using $\hat{\boldsymbol{\pi}}_{CI_{j|\ell-1}}$. The Jacobians are defined as [29, 34]:

$$\mathbf{H}_{\mathbf{f}_{ij}} = \mathbf{J}_{ij} {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1} {}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1} \tag{3.33}$$

$$\mathbf{H}_{\boldsymbol{\pi}_{ij}} = \mathbf{H}_{\mathbf{f}_{ij}} \begin{bmatrix} \lfloor ({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1}) \times \rfloor & -\mathbf{I}_3 \end{bmatrix} \tag{3.34}$$

$$\mathbf{H}_{C_{ij}} = \mathbf{J}_{ij} \begin{bmatrix} {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1} \lfloor {}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1} ({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1}) \times \rfloor & \mathbf{I}_3 \end{bmatrix} \tag{3.35}$$

The measurement Jacobian, $\mathbf{J}_{ij}$ is:

$$\mathbf{J}_{ij} = \left. \frac{\partial \mathbf{h}(\mathbf{f}_i)}{\partial \mathbf{f}} \right|_{\mathbf{f}={}^{C_j}\hat{\mathbf{p}}_{f_i}} = \frac{1}{{}^{C_j}\hat{z}_{f_i}} \begin{bmatrix} 1 & 0 & -\frac{{}^{C_j}\hat{x}_{f_i}}{{}^{C_j}\hat{z}_{f_i}} \\ 0 & 1 & -\frac{{}^{C_j}\hat{y}_{f_i}}{{}^{C_j}\hat{z}_{f_i}} \end{bmatrix} \tag{3.36}$$

where the estimate of the feature in the camera frame is:

$$^{C_j}\hat{\mathbf{p}}_{f_i} = \begin{bmatrix} ^{C_j}\hat{x}_{f_i} \\ ^{C_j}\hat{y}_{f_i} \\ ^{C_j}\hat{z}_{f_i} \end{bmatrix} = {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1}\,{}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1}(^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1}) + {}^{C}\hat{\mathbf{p}}_{I} \tag{3.37}$$

Formulation of Constraints

The residual given in equation 3.32 can not be used for an EKF update because the noise vector $\mathbf{n}_{ij}$ and the state $\tilde{\mathbf{x}}_{\ell|\ell-1}$ are not independent [29]. Specifically, $^{G}\tilde{\mathbf{p}}_{f_i}$ is correlated to both $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{n}_{ij}$ because $^{G}\hat{\mathbf{p}}_{f_i}$ is computed as a function of $\hat{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{z}_{ij}$. Thus, $^{G}\tilde{\mathbf{p}}_{f_i}$ must be removed from the residual equations by the following procedure from [29].

From the N observations of feature f_i, N residual equations are formed (i.e. the jth residual is given in equation 3.32). All N residual equations are combined to form the vector:

$$\begin{aligned} \mathbf{r}_i \simeq & \mathbf{H}_{\boldsymbol{\pi}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{H}_{C_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{\ell|\ell-1} + \\ & \mathbf{H}_{\mathbf{f}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\,{}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_i \end{aligned} \tag{3.38}$$

where $\mathbf{r}_i$ and $\mathbf{n}_i$ are block vectors with elements $\mathbf{r}_{ij}$ and $\mathbf{n}_{ij}$ respectively. The matrices $\mathbf{H}_{\boldsymbol{\pi}_i}$, $\mathbf{H}_{C_i}$ and $\mathbf{H}_{\mathbf{f}_i}$ are formed of block rows of $\mathbf{H}_{\boldsymbol{\pi}_{ij}}$, $\mathbf{H}_{C_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ respectively. Note that the error terms $\tilde{\mathbf{x}}_{\ell|\ell-1}$ appearing in equation 3.38 contain the error terms $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and $\tilde{\boldsymbol{\pi}}_{CI_{j|\ell-1}}$ that appear in equation 3.32.

Next, the residual $\mathbf{r}_i$ is projected onto the left nullspace of the matrix $\mathbf{H}_{\mathbf{f}_i}$. Specifically, the residual vector $\mathbf{r}_i^0$ is defined as:

$$\mathbf{r}_i^0 = \mathbf{V}_i^T\mathbf{r}_i \tag{3.39}$$

where $\mathbf{V}_i$ is a matrix whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$.[1] Substituting the expression for $\mathbf{r}_i$ given in equation 3.38 into equation 3.39 yields:

[1]The matrix $\mathbf{V}_i$ does not have to be explicitly calculated. See [34] for an efficient method for projecting the residual $\mathbf{r}_i$ onto the left nullspace of the matrix $\mathbf{H}_{\mathbf{f}_i}$

$$\mathbf{r}_i^0 = \mathbf{V}_i^T \mathbf{r}_i \simeq \mathbf{H}_i^0 (\hat{\mathbf{x}}_{\ell|\ell-1}, {}^G\hat{\mathbf{p}}_{f_i}) \tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{n}_i^0 \tag{3.40}$$

where $\mathbf{H}_i^0 = \mathbf{V}_i^T \mathbf{H}_{\pi_i} + \mathbf{V}_i^T \mathbf{H}_{C_i}$ and $\mathbf{n}_i^0 = \mathbf{V}_i^T \mathbf{n}_i$. Now, $\mathbf{r}_i^0$ is independent from errors in the feature locations and can be used for an EKF update. Note that equation 3.40 is a linearized constraint between the all of camera poses that observed feature f_i. These constraints express all the information provided by the measurements $\mathbf{z}_{ij}$ [34, 29].

EKF update

The EKF update for the MSCKF algorithm is the same as for EKF-SLAM algorithms. First, the residuals corresponding to each feature undergo a Mahalanobis gating test [29] in order to remove any outliers from being used in the EKF update. The features that pass the test, are used for the EKF update. The following matrices are formed:

$$\mathbf{r}^0 = \begin{bmatrix} \mathbf{r}_1^0 & \dots & \mathbf{r}_i^0 & \dots & \mathbf{r}_k^0 \end{bmatrix}^T \tag{3.41}$$

$$\mathbf{H}^0 = \begin{bmatrix} \mathbf{H}_1^0 & \dots & \mathbf{H}_i^0 & \dots & \mathbf{H}_k^0 \end{bmatrix}^T \tag{3.42}$$

$$\mathbf{R}^0 = \begin{bmatrix} \mathbf{R}_1^0 & & 0 \\ & \ddots & \\ 0 & & \mathbf{R}_k^0 \end{bmatrix} \tag{3.43}$$

where $1 \leq k \leq n$ are the accepted features which have left the field of view of the camera at time step ℓ.

Finally, $\mathbf{r}^0$, $\mathbf{H}^0$ and $\mathbf{R}^0$ are used to perform a standard EKF observation update [15] as described in section 2.2.2:

$$\hat{\mathbf{x}}_{\ell|\ell} = \hat{\mathbf{x}}_{\ell|\ell-1} + \mathbf{K}_\ell \mathbf{r}^0 \tag{3.44}$$

$$\mathbf{P}_{\ell|\ell} = \mathbf{P}_{\ell|\ell-1} - \mathbf{K}_\ell \mathbf{S}_\ell \mathbf{K}_\ell^T \tag{3.45}$$

where

$$\mathbf{S}_\ell = \mathbf{H}^0 \mathbf{P}_{\ell|\ell-1} \mathbf{H}^{0^T} + \mathbf{R}^0 \tag{3.46}$$

$$\mathbf{K}_\ell = \mathbf{P}_{\ell|\ell-1} \mathbf{H}^{0^T} \mathbf{S}_\ell^{-1} \tag{3.47}$$

3.3 Algorithm Comparison

The key difference between EKF-SLAM algorithms and the MSCKF is the way feature measurements are used. The way the MSCKF uses feature measurements drastically improves its performance, in terms of consistency, observability, computational complexity and accuracy, compared to EKF-SLAM algorithms [29].

Accuracy

In EKF-SLAM algorithms, the feature locations are included in the filter state vector; therefore, the IMU pose and the feature locations are jointly estimated. By nature of the EKF, the probability distribution functions of the feature's locations are incorrectly assumed to be Gaussian. Another disadvantage of EKF-SLAM algorithms is that when too many features are visible, they may not all be included in the sate vector in order to allow the system to operate in real time [47]. This means that EKF-SLAM algorithms may not use all the available measurement information.

The MSCKF makes no Gaussianity assumptions because the feature locations are not included in the filter state vector. Instead, a feature is tracked until it leaves the field of view of the camera; then all the feature measurements are used to form probabilistic constraints on the camera poses from which the feature was observed [29]. These constraints form the residuals used for EKF updates and are independent from the errors in feature locations. The MSCKF is able to use each feature measurement by expressing its information in a residual instead of including the feature location in the state vector.

Observability and Consistency

The consistency of an estimation algorithm is crucial to its accuracy [18]. An estimator is consistent if the estimation errors are zero-mean and if they have a covariance matrix smaller or equal to the one calculated by the filter [18]. A primary cause of inconsistency in EKF-SLAM algorithms is due to incorrect observability properties of the estimator [18]. Observability analysis by Li and Mourikis shows that the non-linear SLAM system has four unobservable states: the (3-D) global translation of the state vector is unobservable and the rotation of the vehicle about the gravity vector (i.e. yaw) is unobservable [29]. These four states should, therefore, be unobservable in the linearized system models used in EKF-SLAM and MSCKF algorithms [29].

Standard EKF-SLAM algorithms and the originally proposed MSCKF [34] all erroneously had the property that yaw was observable [29]. This meant that the estimators underestimated the uncertainty of yaw; thereby yielding inconsistent estimators. An improvement for EKF-based estimator consistency, by [18], proposed evaluating the measurement Jacobian matrix using the feature estimate from the first time the feature was observed rather than using the current best estimate from all the feature's observations. Employing this "first estimate Jacobian" method yielded the correct observability properties and improved the consistency of EKF-based estimators [26]. Furthermore, the MSCKF also includes the camera-to-IMU transformation, an observable state, in the state vector and avoids introducing any variables that may be erroneously observable [29].

Though all the EKF-based estimators were more consistent and accurate when the correct observability properties were achieved using the first estimate Jacobian method, the MSCKF still out performed the EKF-SLAM algorithms in terms of accuracy and consistency [29]. Additionally, the correct observability properties of the MSCKF, despite the measurement Jacobians being computed using slightly less accurate linearization points, results in better accuracy and consistency than the best iterative minimization methods [29].

Simulations and real experiments performed in [29] show that the MSCKF algo-

rithm outperforms EKF-SLAM algorithms (and iterative minimization algorithms) in terms of accuracy and consistency. Simulation results in Figure 3-1 show the normalized estimation error squared (NEES) and the root mean squared error (RMSE) of the IMU pose from the MSCKF and three EKF-SLAM algorithms. Note that the RMSE of the MSCKF is smaller than those of the EKF-SLAM algorithms, which indicates better accuracy. Additionally, the NEES is a measure of consistency; the dimension of the pose error is six, so a consistent estimator should have an NEES of six [29]. The inconsistent estimators have a higher NEES.

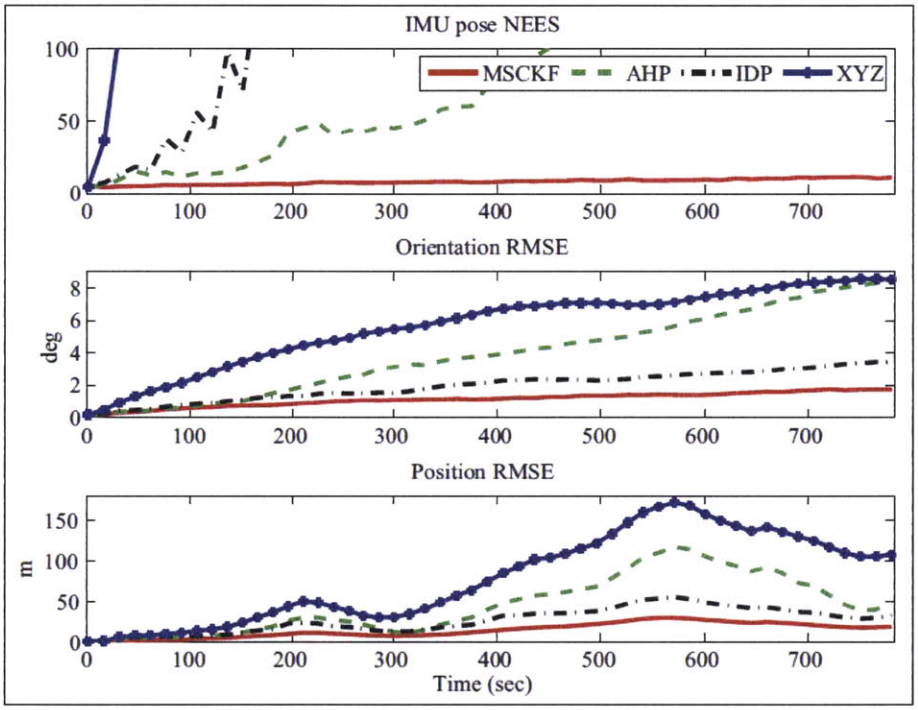


Figure 3-1: The average NEES of the IMU pose and RMSE of the IMU position and orientation. The MSCKF (red line) is compared to three EKF-SLAM methods (green, black and blue lines) [29].

Computational Complexity

The computational complexity for standard EKF-SLAM algorithms is cubic in the number of features in the state vector [47, 29]. The inversion of the mutual information matrix in applying the EKF update (i.e. see equation 3.23) is the dominating computation with a computational cost of $O(n^3)$, where n is the number of features

[47]. This is a huge disadvantage in EKF-SLAM algorithms when an environment has many features (a common situation), because the run-time of the EKF can be unacceptably high [29].

Conversely, the computational complexity of the MSCKF algorithm is linear in the number of features, i.e. $O(n)$. This is because the MSCKF maintains a sliding window of camera poses in the state vector and only uses the feature measurements to impose constraints on the poses [29]. The computational cost of the MSCKF algorithm, however, is driven by the number of camera poses that are included in the state vector; therefore, the maximum computational complexity of the MSCKF is $O(m^3)$, where m is the number of camera poses included in the state vector [34]. The parameter m is chosen based on computational constraints and accuracy requirements. Finally, note that $m << n$ so the computational cost of the MSCKF is much lower than that of EKF-SLAM algorithms; thus making the MSCKF faster than EKF-SLAM [29].

Chapter 4

MSCKF-3D Algorithm

This chapter presents the MSCKF-3D algorithm which is the main contribution of this thesis. The basis for this MSCKF-3D algorithm is the MSCKF algorithm presented in Chapter 3. The MSCKF algorithm is used with monocular VIO systems that are only capable of obtaining 2D feature measurements. The MSCKF-3D algorithm is used with VIO systems capable of obtaining 3D feature measurements. This work specifically uses a monocular camera and a depth sensor to obtain 3D measurements, so the MSCFK-3D algorithm is presented for this type of system. A stereo system, for example, also produces 3D measurements and could use a modified version of the MSCKF-3D algorithm.

The MSCKF-3D algorithm also estimates the time delay between the camera and the IMU. A time offset, t_d, between a camera and IMU means measurements that are simultaneously obtained by the camera and the IMU are time-stamped as if they were taken t_d seconds apart. If time delay is not compensated for, unmodeled errors are introduced into the estimation process that will result in decreased accuracy [28]. The MSCKF-3D algorithm implements a solution proposed by Li and Mourikis to include the time offset in the filter state so it can be estimated in the EKF [28]. This modification could be implemented in all EKF-based VIO algorithms.

4.1 State Parameterization

The IMU state of the MSCKF-3D is the same IMU state used in the MSCKF algorithm (given in section 3.2.1) augmented with the unknown time offset t_d between the camera and IMU time stamps. The IMU state in the MSCKF-3D is the 24×1 vector:

$$\mathbf{x}_{I_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_\ell^T & {}^{G}\mathbf{v}_\ell^T & \mathbf{b}_{\mathbf{g}_\ell}^T & \mathbf{b}_{\mathbf{a}_\ell}^T & {}^{C}_{I}\bar{\mathbf{q}}^T & {}^{C}\mathbf{p}_I^T & t_d \end{bmatrix}^T \tag{4.1}$$

The IMU error state is the 22×1 vector:[1]

$$\tilde{\mathbf{x}}_I = \begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}^T & {}^{G}\tilde{\mathbf{p}}^T & {}^{G}\tilde{\mathbf{v}}^T & \tilde{\mathbf{b}}_{\mathbf{g}}^T & \tilde{\mathbf{b}}_{\mathbf{a}}^T & {}^{C}\tilde{\boldsymbol{\theta}}^T & {}^{C}\tilde{\mathbf{p}}^T & \tilde{t}_d \end{bmatrix}^T \tag{4.2}$$

The complete MSCKF-3D filter state at time step ℓ is the same as the MSCKF filter state as presented in section 3.2.2:

$$\mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^T & \boldsymbol{\pi}_{\ell-1}^T & \boldsymbol{\pi}_{\ell-2}^T & \cdots & \boldsymbol{\pi}_{\ell-N}^T \end{bmatrix}^T \tag{4.3}$$

where $\boldsymbol{\pi}_j = \begin{bmatrix} {}^{I_j}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_j^T \end{bmatrix}^T$, for $j = \ell - N, \ldots, \ell - 1$, are the recorded IMU poses at the times the last N images were recorded.

4.2 Discrete-Time Propagation

The propagation step of the MSCFK-3D algorithm follows the same procedure as the MSCKF algorithm and EKF-SLAM algorithms but with the state transition matrix $\boldsymbol{\Phi}_{I_{\ell-1}}$ extended to include the camera-to-IMU transformation, $\boldsymbol{\pi}_{CI}$ and time offset, t_d. Since neither estimate of $\boldsymbol{\pi}_{CI}$ nor t_d changes during propagation, the state transition matrix is:

[1]The orientation errors ${}^{I_\ell}_{G}\bar{\mathbf{q}}^T$ and ${}^{C}_{I}\bar{\mathbf{q}}^T$ are given in minimal representations ${}^{G}\tilde{\boldsymbol{\theta}}^T$ and ${}^{C}\tilde{\boldsymbol{\theta}}^T$ both $\in \mathbb{R}^3$ [29] so the error state is two dimensions less than its state.

$$\mathbf{\Phi}_{I_{\ell-1}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{qb_g} & \mathbf{0}_3 & \mathbf{0}_{3\times7} \\ \mathbf{\Phi}_{pq} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{\Phi}_{pb_g} & \mathbf{\Phi}_{pa} & \mathbf{0}_{3\times7} \\ \mathbf{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{vb_g} & \mathbf{\Phi}_{va} & \mathbf{0}_{3\times7} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times7} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3\times7} \\ \mathbf{0}_{7\times3} & \mathbf{0}_{7\times3} & \mathbf{0}_{7\times3} & \mathbf{0}_{7\times3} & \mathbf{0}_{7\times3} & \mathbf{I}_7 \end{bmatrix} \tag{4.4}$$

The procedure for discrete-time propagation presented in section 3.1.3 can be followed using the state transition matrix $\mathbf{\Phi}_{I_{\ell-1}}$ given in equation 4.4.

4.3 Observation Update

As in the MSCKF algorithm, the observation updates in the MSCKF-3D algorithm occur when a feature leaves the field of view of the camera. The first step in the observation update is to estimate the global location of the feature using all of the feature observations. Since the measurement system used in this thesis consists of a monocular camera and a depth sensor, each feature observation is a 3D measurement. The jth observation of feature f_i observed from camera pose C_j is:

$$\mathbf{z}_{ij} = \mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i) + \mathbf{n}_{ij} = \begin{bmatrix} \frac{C_j x_{f_i}}{C_j z_{f_i}} \\ \frac{C_j y_{f_i}}{C_j z_{f_i}} \\ C_j z_{f_i} \end{bmatrix} + \mathbf{n}_{ij} \tag{4.5}$$

where $\mathbf{n}_{ij}$ is the measurement noise vector, modeled as zero-mean Gaussian with covariance matrix $\mathbf{R}_{ij} \in \mathbb{R}_{3\times3}$.

The MSCKF only receives 2D feature observations so the goal of the MSCKF-3D is to extend the MSCKF to use the 3D observations. This section will present a model proposed by Dryanovski et al. that is used to estimate the location and uncertainty of each feature observation. Then the global feature location can be estimated using each observation location and its corresponding uncertainty in a weighted least squares estimator. Subsequently, the feature residuals are calculated

and formed into constraints that are used to perform an EKF update.

4.3.1 Feature Location Uncertainty Model

This section presents a method proposed by Dryanovski et al. for estimating the uncertainty of the feature location output by an RGB-D camera [11]. Consider the jth observation of a feature f_i is located at pixel $\mathbf{q}_j = \begin{bmatrix} u_j & v_j & d_j \end{bmatrix}^T$ where u_j and v_j are image coordinates (in pixels) and d_j is the depth measurement obtained from an RGB-D camera. The position of the feature in the camera frame is [11]:

$$ {}^{C_j}\mathbf{p}_{f_i} = \begin{bmatrix} {}^{C_j}x_{f_i} \\ {}^{C_j}y_{f_i} \\ {}^{C_j}z_{f_i} \end{bmatrix} = \begin{bmatrix} \frac{{}^{C_j}z_{f_i}}{f_x}(u_j - c_x) \\ \frac{{}^{C_j}z_{f_i}}{f_y}(v_j - c_y) \\ d_j \end{bmatrix} \tag{4.6} $$

where f_x and f_y are the focal distances of the camera. c_x and c_y are the image optical center. Dryanovski et al. represent the feature position ${}^{C_j}\mathbf{p}_{f_i}$ as Gaussian mixture model in order to estimate the depth uncertainty.

Depth Uncertainty Model

The first step to creating the Gaussian mixture model for depth uncertainty is to treat d_j as a random variable with mean μ_{d_j} and with standard deviation $\sigma_{d_j} \sim \mu_{d_j}^2$ as experimentally determined by Khoshelam and Elberink [20]. Also assume the feature locations u_j and v_j, which are detected from a feature locator, are independent random variables distributed according to a normal distribution $N(\mu_{u_j}, \sigma_{u_j})$ and $N(\mu_{v_j}, \sigma_{v_j})$. Let σ_{u_j} and σ_{v_j} inform the approximate Gaussian kernel:

$$ \mathbf{W} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.7} $$

The second assumption is that the depth uncertainty of the feature depends on the depth reading of its pixel and its surrounding pixels. Thus, assuming ${}^{C_j}z_{f_i}$ is normally distributed, the random variable ${}^{C_j}\hat{z}_{f_i}$, is defined as a mixture of the ${}^{C_j}z_{f_i}$

variables in the local window $\{k :\in [\mu_{u_j} - 1, \mu_{u_j} + 1], \ell :\in [\mu_{v_j} - 1, \mu_{v_j} + 1]\}$. The mean and covariance of the Gaussian mixture are [11]:

$$\hat{\mu}_{z_j} = \sum_{k,\ell} w_{k\ell} \mu_{d_{k\ell}} \tag{4.8}$$

$$\hat{\sigma}^2_{z_j} = \sum_{k,\ell} w_{k\ell} (\sigma^2_{d_{k\ell}} + \mu^2_{d_{k\ell}}) - \hat{\mu}^2_{z_j} \tag{4.9}$$

where the weights of the mixture, $w_{k\ell}$ are from the kernel $\mathbf{W}$.

Discussion of the Gaussian Mixture Model

Dryanovski et al. show that the Gaussian mixture model predicts the uncertainty of feature depth better than a simple model that assumes no uncertainty in u_j and v_j measurements [11]. Allowing for uncertainty in u_j and v_j results in estimating feature depth using a "local window" of pixels' depth measurements rather than a depth measurement of a single pixel. Using a "depth window" for estimation especially improves a feature's predicted depth and its uncertainty when the feature is around an object edge [11]. This is because the measurements of the edge pixels produced by the RGB-D camera tend to jump from background to foreground [11]. This means that only measuring a single pixel located on the edge of an object will tend to produce a much lower predicted uncertainty than the true uncertainty. By comparing the measurements of surrounding pixels, the feature depth uncertainty is better estimated.

3D Uncertainty Model

The mean $\hat{\mu}_{z_j}$ of the Gaussian mixture ${}^{C_j}\hat{z}_{f_i}$ is then used to calculate the means μ_{x_j} and μ_{y_j} of ${}^{C_j}x_{f_i}$ and ${}^{C_j}y_{f_i}$ respectively.

$$\mu_{x_j} = \frac{\hat{\mu}_{z_j}}{f_x}(\mu_{u_j} - c_x) \tag{4.10}$$

$$\mu_{y_j} = \frac{\hat{\mu}_{z_j}}{f_y}(\mu_{v_j} - c_y) \tag{4.11}$$

The 3D uncertainty $\mathbf{\Sigma}_j$ of a point ${}^{C_j}\mathbf{p}_{f_i}$ is estimated as [11]:

$$\mathbf{\Sigma}_j = \begin{bmatrix} \sigma_{x_j}^2 & \sigma_{xy_j} & \sigma_{xz_j} \\ \sigma_{yx_j} & \sigma_{y_j}^2 & \sigma_{yz_j} \\ \sigma_{zx_j} & \sigma_{zy_j} & \hat{\sigma}_{z_j}^2 \end{bmatrix} \tag{4.12}$$

where

$$\begin{aligned} \sigma_{x_j}^2 &= \frac{\hat{\sigma}_{z_j}^2(\mu_{u_j} - c_x)^2 + \sigma_{u_j}^2(\hat{\mu}_{z_j}^2 + \hat{\sigma}_{z_j}^2)}{f_x^2} \\ \sigma_{y_j}^2 &= \frac{\hat{\sigma}_{z_j}^2(\mu_{v_j} - c_y)^2 + \sigma_{v_j}^2(\hat{\mu}_{z_j}^2 + \hat{\sigma}_{z_j}^2)}{f_y^2} \\ \sigma_{xz_j} = \sigma_{zx_j} &= \frac{\hat{\sigma}_{z_j}^2(\mu_{u_j} - c_x)}{f_x} \\ \sigma_{yz_j} = \sigma_{zy_j} &= \frac{\hat{\sigma}_{z_j}^2(\mu_{v_j} - c_y)}{f_y} \\ \sigma_{xy_j} = \sigma_{yx_j} &= \frac{\hat{\sigma}_{z_j}^2(\mu_{u_j} - c_x)(\mu_{v_j} - c_y)}{f_x f_y} \end{aligned} \tag{4.13}$$

Now, the jth observation of the feature has camera coordinates ${}^{C_j}\mathbf{p}_{f_i}$, which is approximated as a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_j = \begin{bmatrix} \mu_{x_j} & \mu_{y_j} & \hat{\mu}_{z_j} \end{bmatrix}^T$ and with uncertainty $\mathbf{\Sigma}_j$

4.3.2 Estimation of Feature Location

When a feature f_i leaves the field of view of the camera at time step ℓ there are N observations of the feature. The set of observations of feature f_i in camera coordinates is ${}^{C}\mathbf{p}_{f_i} = \{{}^{C_{\ell-N}}\mathbf{p}_{f_i}, \ldots, {}^{C_{\ell-1}}\mathbf{p}_{f_i}\}$. Following the procedure from section 4.3.1, ${}^{C}\mathbf{p}_{f_i}$ can be approximated as a set of multivariate Gaussian distributions with means $\boldsymbol{\mu} = \{\boldsymbol{\mu}_{\ell-N}, \ldots, \boldsymbol{\mu}_{\ell-1}\}$ and covariances $\mathbf{\Sigma} = \{\mathbf{\Sigma}_{\ell-N}, \ldots, \mathbf{\Sigma}_{\ell-1}\}$. The goal of this section is to find the best estimate of the feature in global coordinates ${}^{G}\hat{\mathbf{p}}_{f_i}$ using $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$.

Consider the jth observation of the feature with camera coordinates ${}^{C_j}\mathbf{p}_{f_i}$, which is approximated as a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_j = \begin{bmatrix} \mu_{x_j} & \mu_{y_j} & \hat{\mu}_{z_j} \end{bmatrix}^T$ with uncertainty $\mathbf{\Sigma}_j$. The feature position ${}^{C_j}\mathbf{p}_{f_i}$ for $j = \ell - N, \ldots, \ell - 1$ is related to its position in the global frame by [29]:

$$ {}^{C_j}\mathbf{p}_{f_i} = {}^{C}_{I}\mathbf{R}_{j|\ell-1}\,{}^{I}_{G}\mathbf{R}_{j|\ell-1}({}^{G}\mathbf{p}_{f_i} - {}^{G}\mathbf{p}_{j|\ell-1}) + {}^{C}\mathbf{p}_{I} \tag{4.14} $$

The N observations of the feature can be stacked to form the system:

$$ \begin{bmatrix} {}^{C_{\ell-N}}\mathbf{p}_{f_i} \\ \vdots \\ {}^{C_{\ell-1}}\mathbf{p}_{f_i} \end{bmatrix} = \begin{bmatrix} {}^{C}_{I}\mathbf{R}_{\ell-N|\ell-1}\,{}^{I}_{G}\mathbf{R}_{\ell-N|\ell-1} \\ \vdots \\ {}^{C}_{I}\mathbf{R}_{\ell-1|\ell-1}\,{}^{I}_{G}\mathbf{R}_{\ell-1|\ell-1} \end{bmatrix} \begin{bmatrix} {}^{G}\mathbf{p}_{f_i} \end{bmatrix} + \begin{bmatrix} {}^{C}\mathbf{p}_{I} - {}^{C}_{I}\mathbf{R}_{\ell-N|\ell-1}\,{}^{I}_{G}\mathbf{R}^{G}_{\ell-N|\ell-1}\mathbf{p}_{\ell-N|\ell-1} \\ \vdots \\ {}^{C}\mathbf{p}_{I} - {}^{C}_{I}\mathbf{R}_{\ell-1|\ell-1}\,{}^{I}_{G}\mathbf{R}^{G}_{\ell-1|\ell-1}\mathbf{p}_{\ell-1|\ell-1} \end{bmatrix} \tag{4.15} $$

The best estimate for the global coordinates of the feature, ${}^{G}\hat{\mathbf{p}}_{f_i}$ is calculated using a weighted least squares estimate [15].

$$ {}^{G}\hat{\mathbf{p}}_{f_i} = (\mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{m}_y) \tag{4.16} $$

where

$$ \begin{aligned} \mathbf{A} &= \begin{bmatrix} {}^{C}_{I}\mathbf{R}_{\ell-N|\ell-1}\,{}^{I}_{G}\mathbf{R}_{\ell-N|\ell-1} \\ \vdots \\ {}^{C}_{I}\mathbf{R}_{\ell-1|\ell-1}\,{}^{I}_{G}\mathbf{R}_{\ell-1|\ell-1} \end{bmatrix} \\ \mathbf{m}_y &= \begin{bmatrix} {}^{C}\mathbf{p}_{I} - {}^{C}_{I}\mathbf{R}_{\ell-N|\ell-1}\,{}^{I}_{G}\mathbf{R}^{G}_{\ell-N|\ell-1}\mathbf{p}_{\ell-N|\ell-1} \\ \vdots \\ {}^{C}\mathbf{p}_{I} - {}^{C}_{I}\mathbf{R}_{\ell-1|\ell-1}\,{}^{I}_{G}\mathbf{R}^{G}_{\ell-1|\ell-1}\mathbf{p}_{\ell-1|\ell-1} \end{bmatrix} \end{aligned} \tag{4.17} $$

Since ${}^{C_j}\mathbf{p}_{f_i}$ is approximated as a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_j$ and with uncertainty $\mathbf{\Sigma}_j$, the measurements are described by:

$$ \mathbf{y} = \begin{bmatrix} {}^{C_{\ell-N}}\mathbf{p}_{f_i} \\ \vdots \\ {}^{C_{\ell-1}}\mathbf{p}_{f_i} \end{bmatrix} \simeq \begin{bmatrix} \boldsymbol{\mu}_{\ell-N} \\ \vdots \\ \boldsymbol{\mu}_{\ell-1} \end{bmatrix} \tag{4.18} $$

with covariance matrix:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{\ell-N} & & \\ & \ddots & \\ & & \mathbf{\Sigma}_{\ell-1} \end{bmatrix} \tag{4.19}$$

4.3.3 Feature Residual Calculation

The general procedure for calculating the feature residuals in the MSCKF algorithm presented in section 3.2.4 can be followed to calculate the feature residuals for the MSCKF-3D. However, the augmentation of the IMU state with time delay, t_d, and the new measurement model, $\mathbf{h}(\mathbf{x}_{I_j}, \mathbf{f}_i)$, introduce modifications for calculating the feature residuals.

The residual (for $j = \ell - N, \ldots, \ell - 1$) between the actual and expected feature measurement is still:

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\boldsymbol{\pi}}_{j|\ell-1}, \hat{\boldsymbol{\pi}}_{CI_{j|\ell-1}}, {}^{G}\hat{\mathbf{p}}_{f_i}) \tag{4.20}$$

The linearized approximation for the MSCKF-3D becomes [28]:

$$\mathbf{r}_{ij} \simeq \mathbf{H}_{\boldsymbol{\pi}_{ij}} \tilde{\boldsymbol{\pi}}_{j|\ell-1} + \mathbf{H}_{C_{ij}} \tilde{\boldsymbol{\pi}}_{CI_{j|\ell-1}} + \mathbf{H}_{\mathbf{f}_{ij}} {}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{H}_{t_{ij}} \tilde{t}_d + \mathbf{n}_{ij} \tag{4.21}$$

The term $\mathbf{H}_{t_{ij}} \tilde{t}_d$ appears in this linearized residual because the measurement function $\mathbf{h}$ is a function of feature position, which is a function of the time delay. $\mathbf{H}_{t_{ij}}$ is defined as [28]:

$$\mathbf{H}_{t_{ij}} = \mathbf{J}_{ij} {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1} [\lfloor {}^{I}\boldsymbol{\omega} \times \rfloor {}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1} ({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1}) - {}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1} {}^{G}\hat{\mathbf{v}}_{I}] \tag{4.22}$$

Since the MSCKF-3D has a 3D measurement of feature location (equation 4.5), the measurement Jacobian, $\mathbf{J}_{ij}$, becomes:

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{h}(\mathbf{f}_i)}{\partial \mathbf{f}}\bigg|_{\mathbf{f}={}^{C_j}\hat{\mathbf{p}}_{f_i}} = \begin{bmatrix} \frac{1}{{}^{C_j}\hat{z}_{f_i}} & 0 & -\frac{{}^{C_j}\hat{x}_{f_i}}{{}^{C_j}\hat{z}^2_{f_i}} \\ 0 & \frac{1}{{}^{C_j}\hat{z}_{f_i}} & -\frac{{}^{C_j}\hat{y}_{f_i}}{{}^{C_j}\hat{z}^2_{f_i}} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

where the estimate of the feature in the camera frame is still:

$${}^{C_j}\hat{\mathbf{p}}_{f_i} = \begin{bmatrix} {}^{C_j}\hat{x}_{f_i} \\ {}^{C_j}\hat{y}_{f_i} \\ {}^{C_j}\hat{z}_{f_i} \end{bmatrix} = {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1}{}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1}({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1}) + {}^{C}\hat{\mathbf{p}}_{I} \tag{4.24}$$

The equations for the Jacobians $\mathbf{H}_{\boldsymbol{\pi}_{ij}}$, $\mathbf{H}_{\mathbf{f}_{ij}}$ and $\mathbf{H}_{C_{ij}}$ are unchanged from the MSCKF case are defined as [29, 34]:

$$\begin{aligned} \mathbf{H}_{\mathbf{f}_{ij}} &= \mathbf{J}_{ij}{}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1}{}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1} \\ \mathbf{H}_{\boldsymbol{\pi}_{ij}} &= \mathbf{H}_{\mathbf{f}_{ij}}\begin{bmatrix} \lfloor({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1})\times\rfloor & -\mathbf{I}_3 \end{bmatrix} \\ \mathbf{H}_{C_{ij}} &= \mathbf{J}_{ij}\begin{bmatrix} {}^{C}_{I}\hat{\mathbf{R}}_{j|\ell-1}\lfloor{}^{I}_{G}\hat{\mathbf{R}}_{j|\ell-1}({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{j|\ell-1})\times\rfloor & \mathbf{I}_3 \end{bmatrix} \end{aligned} \tag{4.25}$$

4.3.4 Formulation of Constraints

Similarly to the MSCKF algorithm, the residual given in equation 4.21 can not be used for an EKF update in the MSCKF-3D because the noise vector $\mathbf{n}_{ij}$ and the state $\tilde{\mathbf{x}}_{\ell|\ell-1}$ are not independent [29]. Specifically, ${}^{G}\tilde{\mathbf{p}}_{f_i}$ is correlated to both $\tilde{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{n}_{ij}$ because ${}^{G}\hat{\mathbf{p}}_{f_i}$ is computed as a function of $\hat{\boldsymbol{\pi}}_{j|\ell-1}$ and $\mathbf{z}_{ij}$. Thus, ${}^{G}\tilde{\mathbf{p}}_{f_i}$ must be removed from the residual equations using the same procedure as in the MSCKF algorithm [29]. The procedure for forming constraints in the MSCKF-3D is presented here.

First, all N residual equations are combined to form the vector:

$$\begin{aligned} \mathbf{r}_i \simeq \mathbf{H}_{\boldsymbol{\pi}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{H}_{C_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{\ell|\ell-1} + \\ \mathbf{H}_{t_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{H}_{\mathbf{f}_i}(\hat{\mathbf{x}}_{\ell|\ell-1}, {}^{G}\hat{\mathbf{p}}_{f_i}){}^{G}\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_i \end{aligned} \tag{4.26}$$

where $\mathbf{r}_i$ and $\mathbf{n}_i$ are block vectors with elements $\mathbf{r}_{ij}$ and $\mathbf{n}_{ij}$ respectively. The

matrices $\mathbf{H}_{\pi_i}$, $\mathbf{H}_{C_i}$, $\mathbf{H}_{t_i}$ and $\mathbf{H}_{\mathbf{f}_i}$ are formed of block rows of $\mathbf{H}_{\pi_{ij}}$, $\mathbf{H}_{C_{ij}}$, $\mathbf{H}_{t_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ respectively.

Next, the residual $\mathbf{r}_i$ is projected onto the left nullspace of the matrix $\mathbf{H}_{\mathbf{f}_i}$. Specifically, the residual vector $\mathbf{r}_i^0$ is defined as:

$$\mathbf{r}_i^0 = \mathbf{V}_i^T \mathbf{r}_i \tag{4.27}$$

where $\mathbf{V}_i$ is a matrix whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$.[2] Substituting the expression for $\mathbf{r}_i$ given in equation 4.26 into equation 4.27 yields:

$$\mathbf{r}_i^0 = \mathbf{V}_i^T \mathbf{r}_i \simeq \mathbf{H}_i^0 (\hat{\mathbf{x}}_{\ell|\ell-1}, {}^G\hat{\mathbf{p}}_{f_i}) \tilde{\mathbf{x}}_{\ell|\ell-1} + \mathbf{n}_i^0 \tag{4.28}$$

where

$$\begin{aligned} \mathbf{H}_i^0 &= \mathbf{V}_i^T \mathbf{H}_{\pi_i} + \mathbf{V}_i^T \mathbf{H}_{C_i} + \mathbf{V}_i^T \mathbf{H}_{t_i} \\ \mathbf{n}_i^0 &= \mathbf{V}_i^T \mathbf{n}_i \end{aligned} \tag{4.29}$$

Now, $\mathbf{r}_i^0$ is independent from errors in the feature locations and can be used for an EKF update. Note that equation 4.28 is a linearized constraint between the all of camera poses that observed feature f_i. These constraints express all the information provided by the measurements $\mathbf{z}_{ij}$ [34, 29].

4.3.5 EKF Update

With the MSCKF-3D definitions of $\mathbf{r}_i^0$ and $\mathbf{H}_i^0$ given in equations 4.28, the EKF update for the MSCKF-3D algorithm is the same as for the MSCKF algorithm. First, the residuals corresponding to each feature undergo a Mahalanobis gating test [29] in order to remove any outliers from being used in the EKF update. The features that pass the test, are used for the EKF update. The following matrices are formed:

$$\mathbf{r}^0 = \begin{bmatrix} \mathbf{r}_1^0 & \dots & \mathbf{r}_i^0 & \dots & \mathbf{r}_k^0 \end{bmatrix}^T \tag{4.30}$$

[2] The matrix $\mathbf{V}_i$ does not have to be explicitly calculated. See [34] for an efficient method for projecting the residual $\mathbf{r}_i$ onto the left nullspace of the matrix $\mathbf{H}_{\mathbf{f}_i}$

$$\mathbf{H}^0 = \begin{bmatrix} \mathbf{H}_1^0 & \dots & \mathbf{H}_i^0 & \dots & \mathbf{H}_k^0 \end{bmatrix}^T \tag{4.31}$$

$$\mathbf{R}^0 = \begin{bmatrix} \mathbf{R}_1^0 & & 0 \\ & \ddots & \\ 0 & & \mathbf{R}_k^0 \end{bmatrix} \tag{4.32}$$

where $1 \leq k \leq n$ are the accepted features which have left the field of view of the camera at time step ℓ.

Finally, $\mathbf{r}^0$, $\mathbf{H}^0$ and $\mathbf{R}^0$ are used to perform a standard EKF update [15]:

$$\hat{\mathbf{x}}_{\ell|\ell} = \hat{\mathbf{x}}_{\ell|\ell-1} + \mathbf{K}_\ell \mathbf{r}^0 \tag{4.33}$$

$$\mathbf{P}_{\ell|\ell} = \mathbf{P}_{\ell|\ell-1} - \mathbf{K}_\ell \mathbf{S}_\ell \mathbf{K}_\ell^T \tag{4.34}$$

where

$$\mathbf{S}_\ell = \mathbf{H}^0 \mathbf{P}_{\ell|\ell-1} \mathbf{H}^{0^T} + \mathbf{R}^0 \tag{4.35}$$

$$\mathbf{K}_\ell = \mathbf{P}_{\ell|\ell-1} \mathbf{H}^{0^T} \mathbf{S}_\ell^{-1} \tag{4.36}$$

4.4 MSCKF-3D Algorithm Summary

The MSCKF-3D algorithm presented in this chapter is for a VIO system consisting of a monocular camera and a depth sensor which produce 3D feature measurements. The basis for this MSCKF-3D algorithm is the MSCKF algorithm, presented in Chapter 3, which is extended to use the 3D feature observations. This is accomplished using a model proposed by Dryanovski et al. that estimates the 3D location and uncertainty of each feature observation by approximating it as a multivariate Gaussian distribution. Then, using all observations of a feature, a weighted least squares estimate of the global feature location is obtained. Additionally, the MSCKF-3D algorithm includes the time offset between the camera and the IMU in the filter state so it can be estimated in the EKF; this technique can be applied to the MSCKF and other EKF-

based VIO algorithms. An overview of the MSCKF-3D algorithm is presented in algorithm 1.

Algorithm 1 Multi-state constraint Kalman Filter (MSCKF)-3D

Initialization:

IMU state $\mathbf{x}_{I_\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_\ell^T & {}^{G}\mathbf{v}_\ell^T & \mathbf{b}_{\mathbf{g}_\ell}^T & \mathbf{b}_{\mathbf{a}_\ell}^T & {}^{C}_{I}\bar{\mathbf{q}}^T & {}^{C}\mathbf{p}_I^T & t_d \end{bmatrix}^T$

Filter state $\mathbf{x}_\ell = \begin{bmatrix} \mathbf{x}_{I_\ell}^T & \boldsymbol{\pi}_{\ell-1}^T & \boldsymbol{\pi}_{\ell-2}^T & \cdots & \boldsymbol{\pi}_{\ell-N}^T \end{bmatrix}^T$

Propagation: Each time the IMU is sampled:

- Propagate state estimate $\hat{\mathbf{x}}_{\ell-1|\ell-1}$ to $\hat{\mathbf{x}}_{\ell|\ell-1}$
- Calculate IMU error state transition matrix $\boldsymbol{\Phi}_{I_{\ell-1}}$
- Propagate error state $\tilde{\mathbf{x}}_{\ell-1|\ell-1}$ to $\tilde{\mathbf{x}}_{\ell|\ell-1}$
- Propagate filter covariance matrix $\mathbf{P}_{\ell-1|\ell-1}$ to $\mathbf{P}_{\ell|\ell-1}$

Update: Each time a new image is recorded:

- Augment $\mathbf{x}_\ell$ with current IMU pose $\boldsymbol{\pi}_\ell$
- Process image: Extract and match features
- For each completed feature track $\mathbf{f}_i$:
  - Approximate each feature observation $\mathbf{f}_{ij}$ as a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_j = \begin{bmatrix} \mu_{x_j} & \mu_{y_j} & \hat{\mu}_{z_j} \end{bmatrix}^T$ and uncertainty $\boldsymbol{\Sigma}_j$
  - Estimate ${}^{G}\hat{\mathbf{p}}_{f_i}$ using weighted least squares
  - Calculate each feature observation residual $\mathbf{r}_{ij}$
  - Determine $\mathbf{r}_i^0$, the constraint between all camera poses that observed $\mathbf{f}_i$
  - Perform Mahalanobis gating test
- Perform an EKF update with all accepted features to find $\hat{\mathbf{x}}_{\ell|\ell}$ and $\mathbf{P}_{\ell|\ell}$

State Management: Remove from $\mathbf{x}_\ell$ all IMU poses $\boldsymbol{\pi}_j$ for which all associated features have been processed

Chapter 5

Experimental Results and Analysis

The performance of the multi-state constraint Kalman filter (MSCKF) and the MSCKF-3D have been evaluated using real-world data obtained by flying a custom-built quadrotor in an indoor office environment. In this section, the quadrotor system used for data collection is described. The quadrotor state estimates produced by the MSCKF and the MSCKF-3D (obtained by post-processing the experimental data) are then presented and compared to the true quadrotor state. Finally, the performance, in terms of accuracy and computational time, of the MSCKF and the MSCKF-3D are compared.

5.1 Experimental Platform

For indoor navigation of an office environment, a quadrotor has been custom-built and equipped with an IMU, an RGB-D camera and an onboard computer. Size and weight of the components are important factors since the quadrotor needs to be able to support the weight during flight and maneuver through hallways. Additionally, the power consumption of the components will directly affect maximum time of flight of the quadrotor. In addition to the IMU, RGB-D camera and computer, the quadrotor frame supports a motor and a battery. The total weight of the quadrotor is approximately 974g. The quadrotor is shown in figure 5-1 and its components are described in the rest of this section.

Figure 5-1: Custom-built Quadrotor

Inertial Measurement Unit

The low-cost, low-weight IMU chosen for the quadrotor is the MPU-6000 [35]. The IMU is housed in a PX4FMU (which also has a magnetometer and barometer) [39]. Table 5.1 describes the specifications of the IMU, which consists of a gyroscope and an accelerometer.

Table 5.1: MPU-6000 Specifications

IMU	Dimensions (of the PX4FMU)	$5 \times 3.6 \times 0.73$ cm
	Power consumption	< 14 mW
	Sample Rate	200 Hz
Accelerometer	Power spectral density	$400\mu g/\sqrt{Hz}$
	x axis precision	$\pm 50mg$
	y axis precision	$\pm 50mg$
	z axis precision	$\pm 80mg$
Gyroscope	Total RMS noise	$0.05°/s$-rms

RGB-D Camera

The quadrotor is equipped with an Asus Xtion Pro Live camera [2]. This camera produces RGB and depth images in VGA resolution (640 x 480 pixels) at 30 Hz or QVGA resolution (320 x 240 pixels) at 60 Hz. It is designed for indoor operation between 0.8m and 3.5m, making it a suitable choice for data collection in the indoor office environment. It has been stripped of its original packaging in order to keep the size and weight as small as possible. Its dimensions are $16 \times 2 \times 3$ cm and it weighs about 88g. The power consumption is less than 2.5 W.

Onboard Computing System

The computer onboard the quadrotor is the Gigabyte Brix GB-XM1-3537 [16]. This computer has a dual core 2GHz Intel i7 processor with 8GB RAM. The casing from the computer has been removed so it weighs 171 grams and the approximate dimensions are $10 \times 10 \times 3$ cm.

The onboard computer uses the Robot Operating System (ROS)[40] framework to interface with the IMU and the RGB-D camera. ROS is commonly used in robotics applications and has many built in packages and libraries that support writing software for robot operation.

The onboard computer is currently used to manually control the quadrotor and store the data that is output by the IMU and the RGB-D camera. The sensor data is then post-processed (offboard on a desktop computer) using the MSCKF and the MSCKF-3D in MATLAB.

Truth Data

For analysis of the MSCKF and the MSCKF-3D, the quadrotor state estimates from the filters are compared to the true quadrotor trajectory obtained from the AprilTag system [37]. This is a visual fiducial system that allows 6DOF localization of features from a single image. 51 distinct April tags were posted in the hallways where the quadrotor was flown. The tag locations were all surveyed so that their relative posi-

tions in the hallway were known. The AprilTags software post-processes the vision data stored during the quadrotor flight, and detects the tags and determines the true vehicle location based on the known tag locations. For more information on AprilTags and the open source software, see [37].

5.2 Experimental Results

The performance of the MSCKF and the MSCKF-3D have been evaluated experimentally using the custom-built quadrotor. During the experiments, the quadrotor stores the vision data (in VGA resolution) and the inertial data onboard. Then, the data is transferred to a desktop computer to be processed by the MSCKF and the MSCKF-3D. Both filters use the same data to produce state estimates of the quadrotor throughout the entire experiment. The position estimates produced by the filters are compared to the true vehicle location obtained from the AprilTags.

Trajectory Alignment

In order to compare the estimates from both filters, the estimated trajectories are aligned to the true trajectory by minimizing the mean squared error (MSE) of the vehicle trajectory over the first few meters of the trajectory. Note that only the beginning of the estimated trajectory is used for alignment, rather than aligning the entire estimated trajectory, because the estimation errors build up throughout time. The beginning of the trajectory is most accurately estimated, and is therefore used for alignment to the true trajectory.

5.2.1 Walking Datasets

In the following two datasets, the quadrotor was carried through an indoor office. In the first dataset a single rectangular loop through the building was traversed. This is a challenging trajectory for the filters to estimate because the environment is poorly lit and has long regions with blank walls, i.e. few features can be used for vision updates. Examples of these images are shown in figure 5-2.

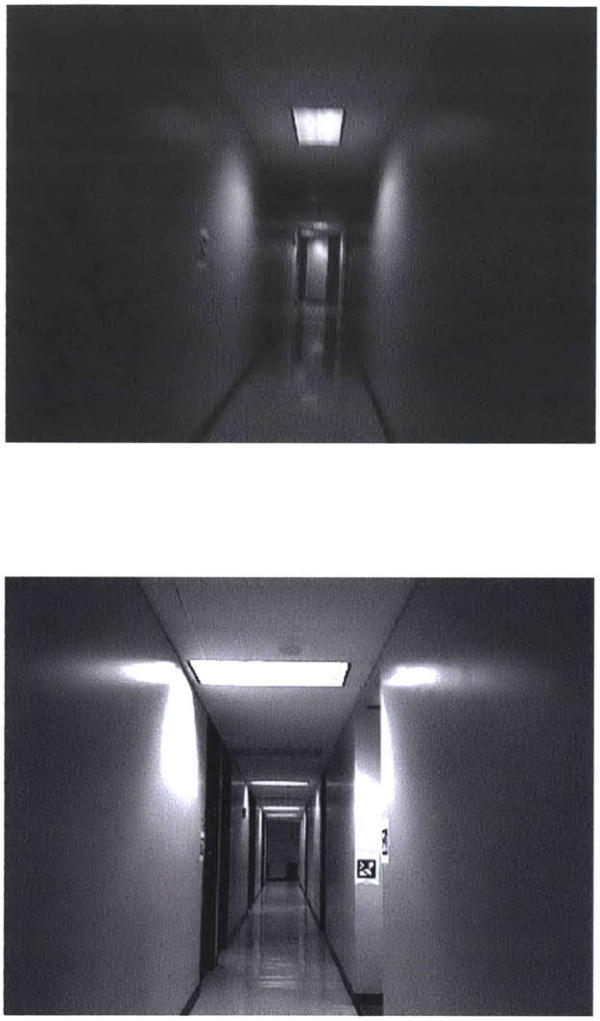


Figure 5-2: Images used for trajectory estimation of the indoor office trajectory. These are example images of the poorly lit hallways with limited features that make trajectory estimation using vision challenging.

The true rectangular loop trajectory, about 100 meters in length, and the estimated trajectories produced by the MSCKF and the MSCKF-3D are shown in figure 5-3. The second dataset followed the same path through the building, however, two rectangularloops were completed. The true and estimated trajectories are shown in figure 5-4. In both datasets the first 15 meters of the estimated trajectories are aligned to the true trajectory as was previously described.

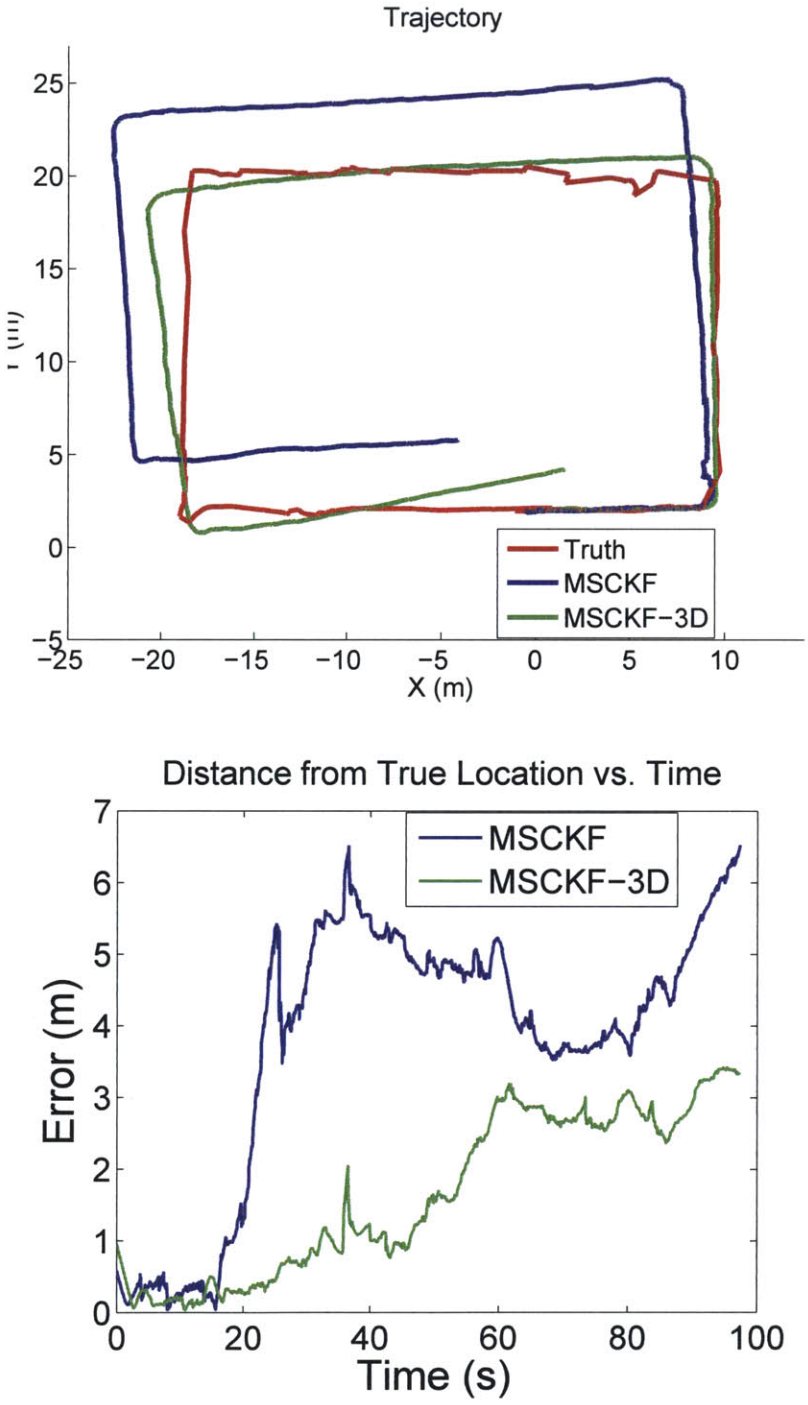


Figure 5-3: (Top) Trajectory obtained by walking the quadrotor around an office building. A single rectangular loop through the hallways is completed (shown in red). The estimated trajectories produced by the MSCKF and the MSCKF-3D are shown in blue and green respectively. (Bottom) Distance between the true location and the estimated location for the MSCKF and the MSCKF-3D.

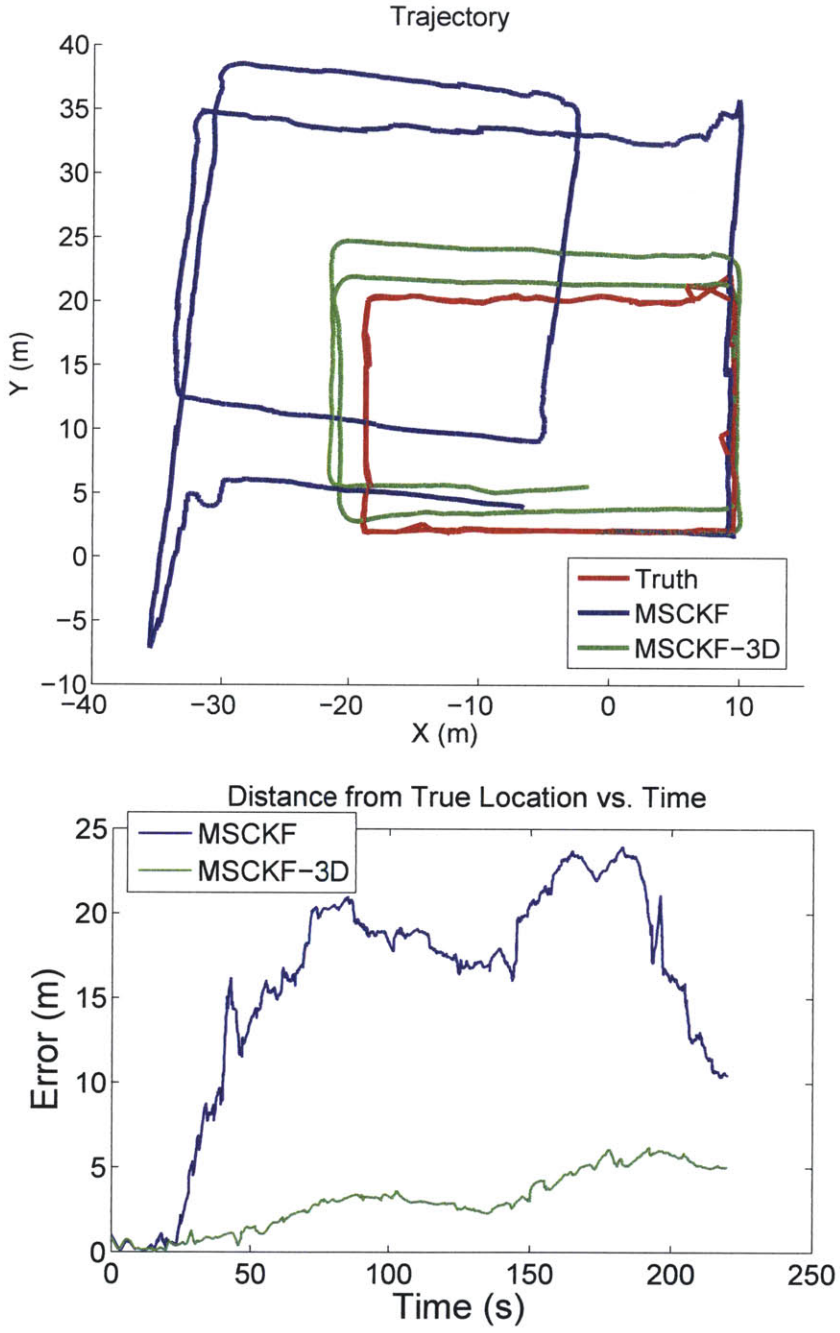


Figure 5-4: (Top) Trajectory obtained by walking the quadrotor around an office building. Two rectangular loops through the hallways are completed (shown in red). The estimated trajectories produced by the MSCKF and the MSCKF-3D are shown in blue and green respectively. (Bottom) Distance between the true location and the estimated location for the MSCKF and the MSCKF-3D.

The results from figures 5-3 and 5-4 demonstrate that the MSCKF-3D outperforms the MSCKF in estimating the true trajectory of the vehicle. Notice the two different size loops estimated by the MSCKF in figure 5-4; this is a result of the MSCKF struggling to estimate movement during the straight portions of the hallways. This is easily explained in figure 5-6, which shows the trajectory estimates and the 2σ uncertainty bounds for both filters in the double-loop trajectory. The x and y position and velocity uncertainties of the MSCKF estimates are much larger than the those of the MSCKF-3D. The depth measurements used in the MSCKF-3D allow the estimator to better predict vehicle velocity and recover a more accurate trajectory estimate through these regions.

Figure 5-6 also shows the error bounds of both filters grow as the quadrotor turns through corners. This makes sense because as the quadrotor is walked through the turns, there are very few features to use for vision updates, as shown in figure 5-5. This makes it very difficult for the filters to estimate the true angle of the turn that the quadrotor makes. Furthermore, since vision and inertial errors accumulate over time, when the angle of the turn is incorrectly estimated, the position error increases as the vehicle continues to move.

Figure 5-5: Image of a turn used for trajectory estimation of the indoor office trajectory. This image has very few features, and makes estimating the angle of the turn using vision very challenging.

Finally, figure 5-6 shows the vertical position estimates of the quadrotor. Though the quadrotor is carried and remains at a nearly constant vertical position, both filters estimate the vertical position is increasing over time. This is a due to the drift of the IMU measurements.

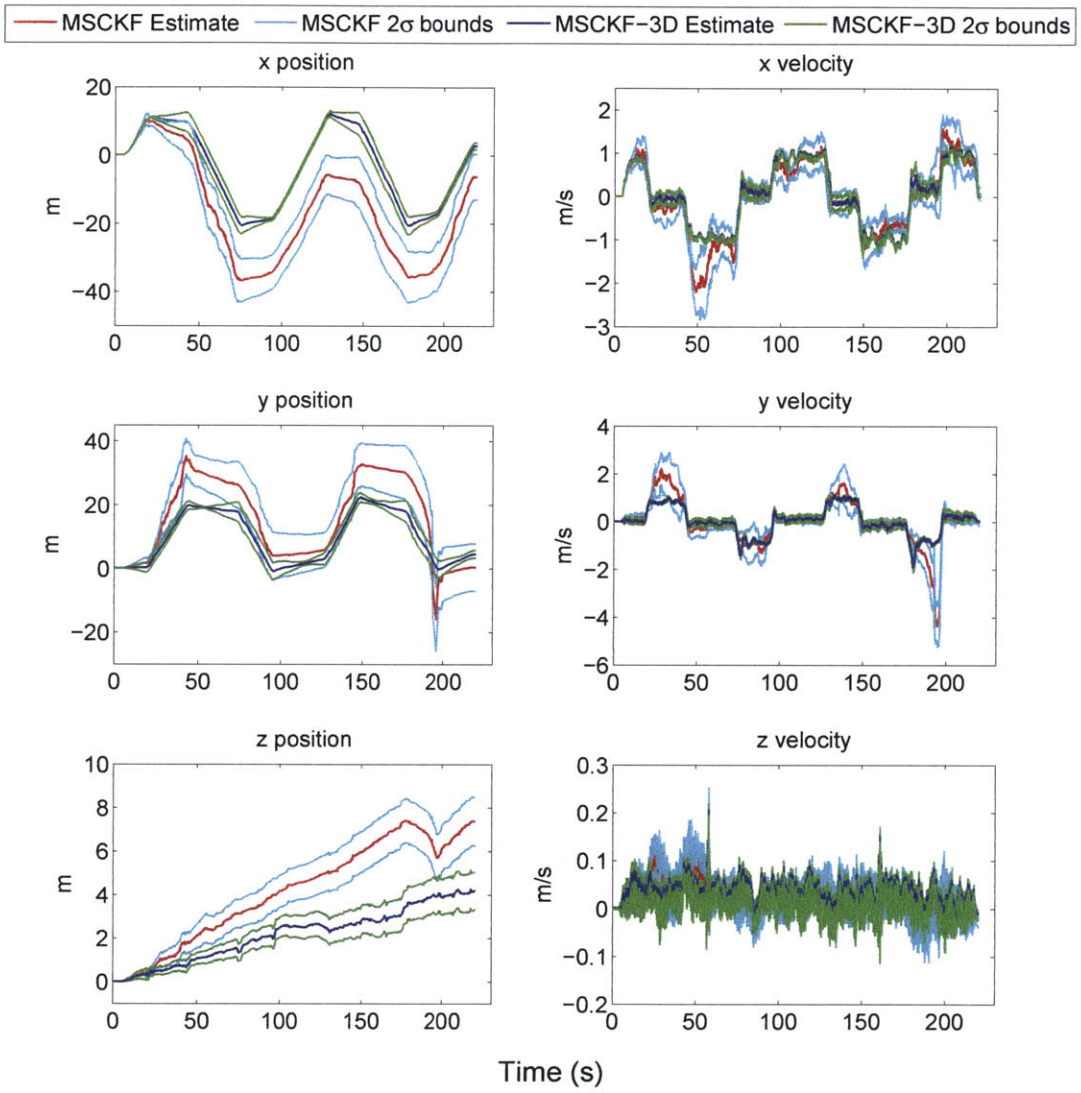


Figure 5-6: Estimation of quadrotor position and velocity during double loop walking trajectory. The blue lines are the MSCKF-3D estimates and the green lines are the corresponding two-sigma error bounds. The red lines are the MSCKF estimates and the cyan lines are the corresponding two-sigma error bounds.

The MSCKF and the MSCKF-3D also estimate vehicle orientation; however, no truth data is available for comparison. To confirm the general correctness, the

MSCKF-3D estimation of vehicle orientation during the double loop trajectory is given in figure 5-7. The MSCKF estimation of vehicle orientation is similar and is not shown here. Since the quadrotor was carried, its pitch and roll remain nearly zero (the roll estimate is 180 degrees because of the orientation of the quadrotor in the global reference frame). The yaw estimate corresponds with the turns through the hallways. Also note that the uncertainty bounds become smaller as the quadrotor moves and more information is available.

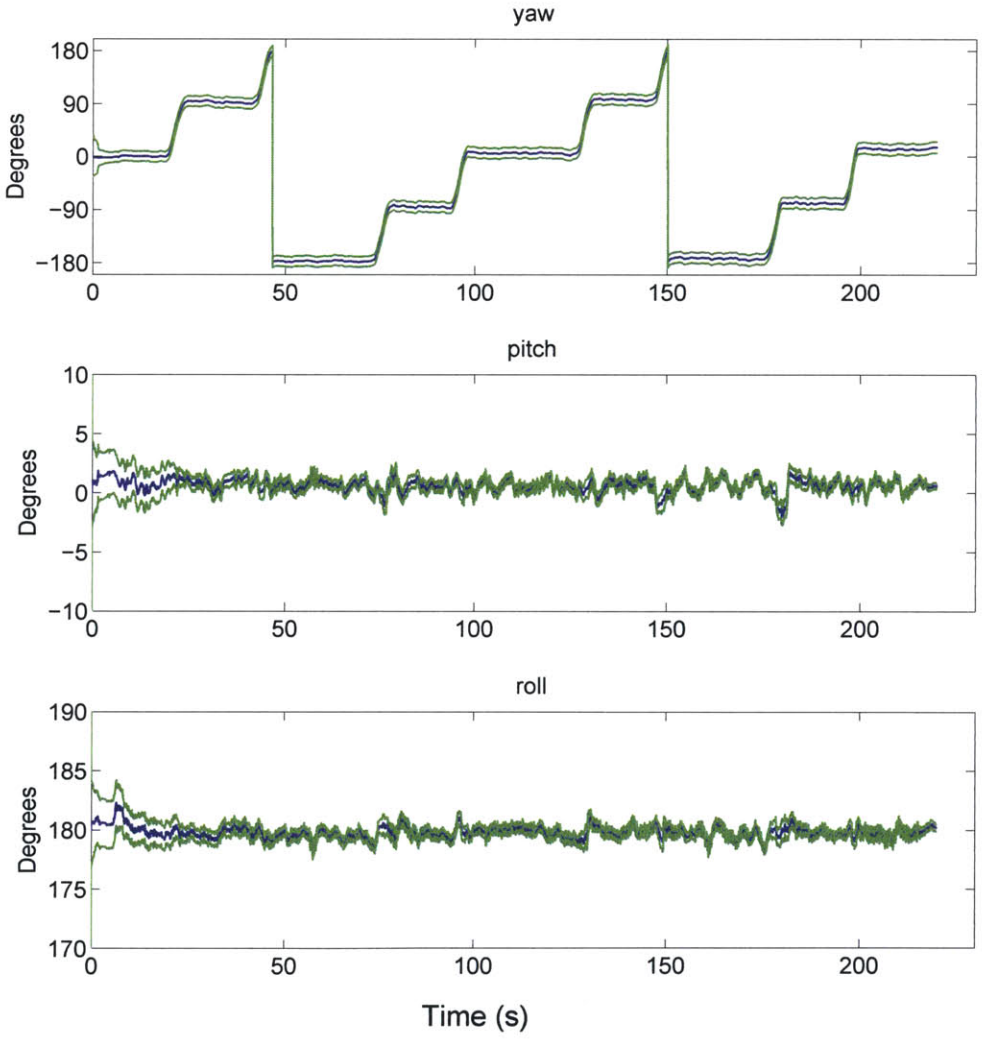


Figure 5-7: MSCKF-3D estimation of quadrotor orientation during double loop walking trajectory. The blue lines are the estimates and the green lines are the two-sigma error bounds.

Figure 5-8 shows the MSCKF-3D estimates of the IMU biases. There is no truth data available for comparison, however, these estimates make sense, as the bias estimates move around during the beginning before settling down. As expected, the bias estimate error bounds are the same order of magnitude for each axis of the gyroscope. Conversely, the vertical axis of the accelerometer has much smaller error bounds than the other two axes because it is aligned with the gravitational axis.

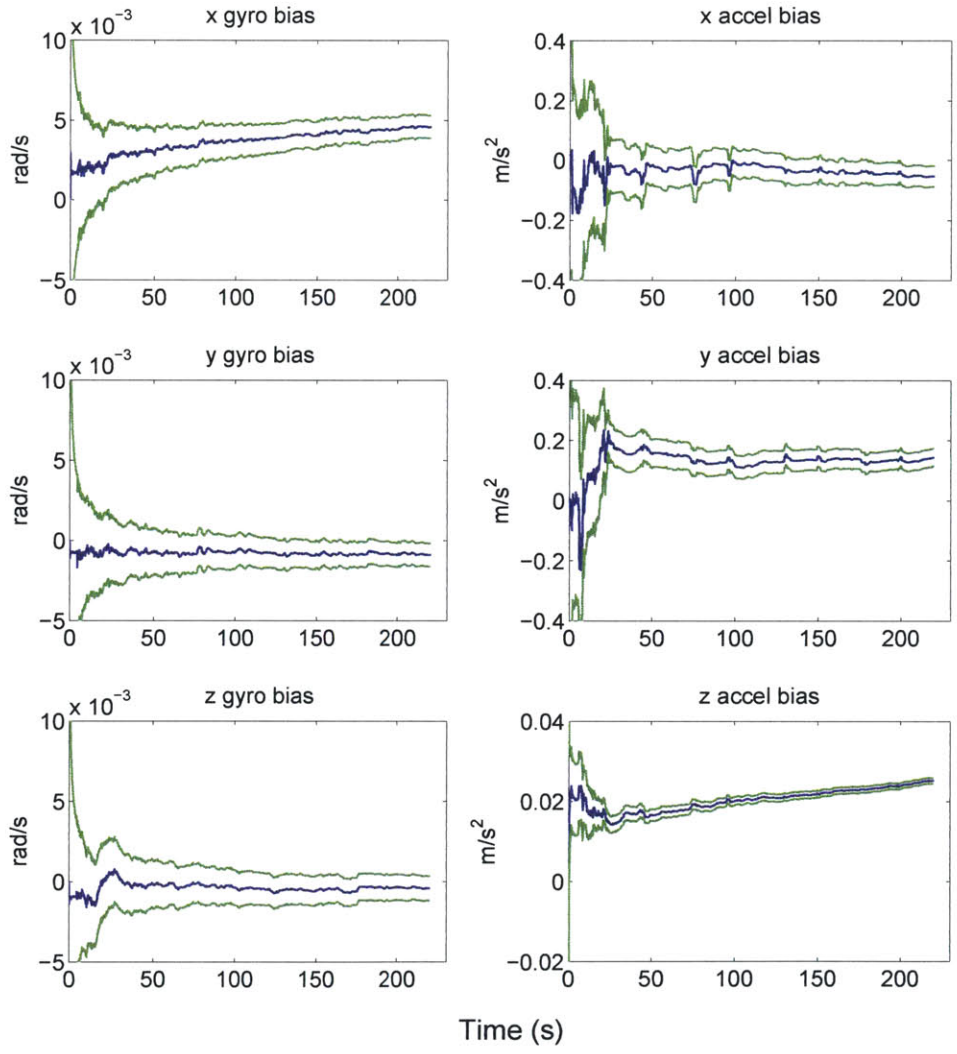


Figure 5-8: MSCKF-3D estimation of IMU biases during double loop walking trajectory. The blue lines are the estimated biases and the green lines are the two-sigma error bounds.

The MSCKF-3D estimation of the transformation between the camera and the IMU is shown in figure 5-9. The IMU coordinate frame in North-East-Down and the camera coordinate frame is East-Down-North. The estimated rotation to get from the camera frame to the IMU frame agrees with these coordinate frames. The translation between the IMU and the focal point of the camera is also estimated; the camera is mounted in front of the IMU, and the estimated 10 cm translation in the x direction is accurate.

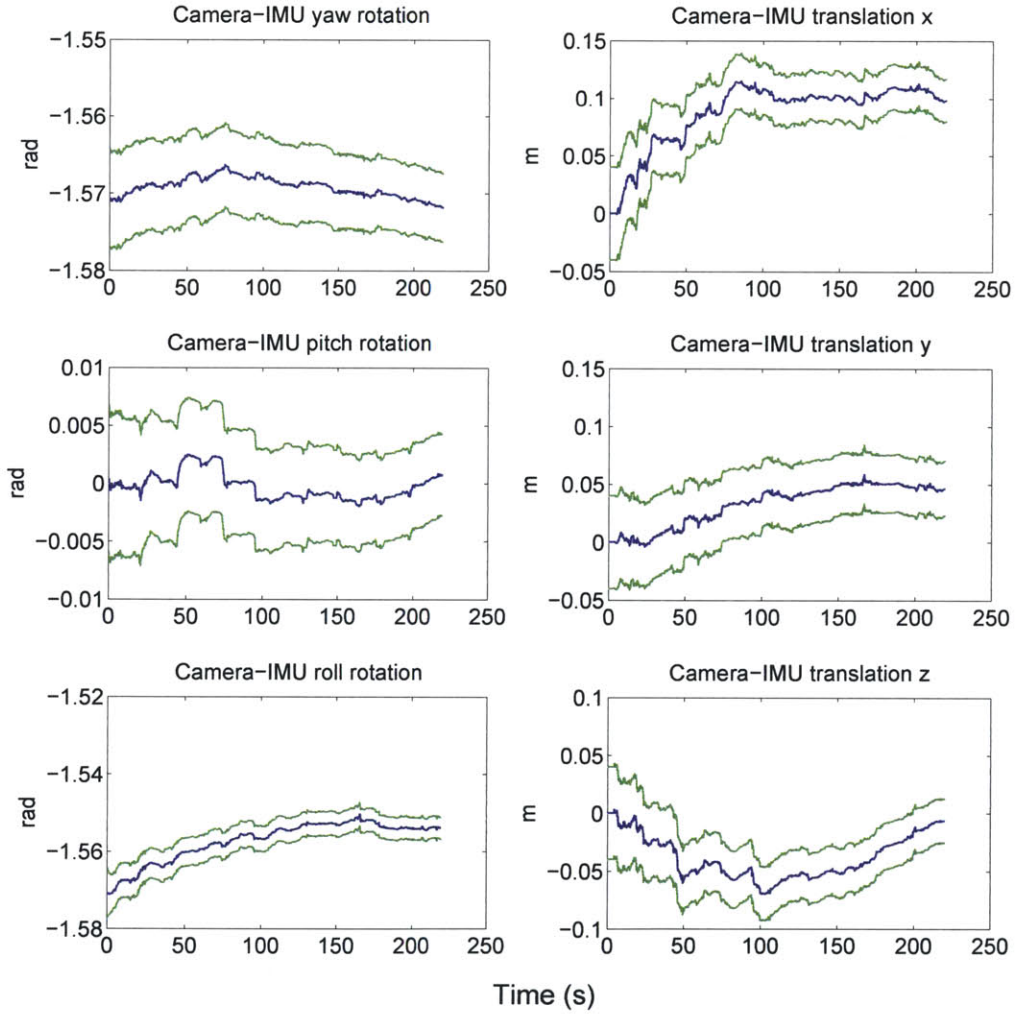


Figure 5-9: MSCKF-3D estimation of camera-IMU transformation during double loop walking trajectory. The blue lines are the estimates and the green lines are the two-sigma error bounds.

Finally, figure 5-10 indicates the MSCKF-3D estimates an approximately 20 ms timing offset between the timstamps on the IMU measurements and the camera measurements. That means that if measurements are received simultaneously from the IMU and the camera, the timestamps indicate the IMU measurement occurred 20 ms before the camera measurement. The MSCKF estimated a similar offset.

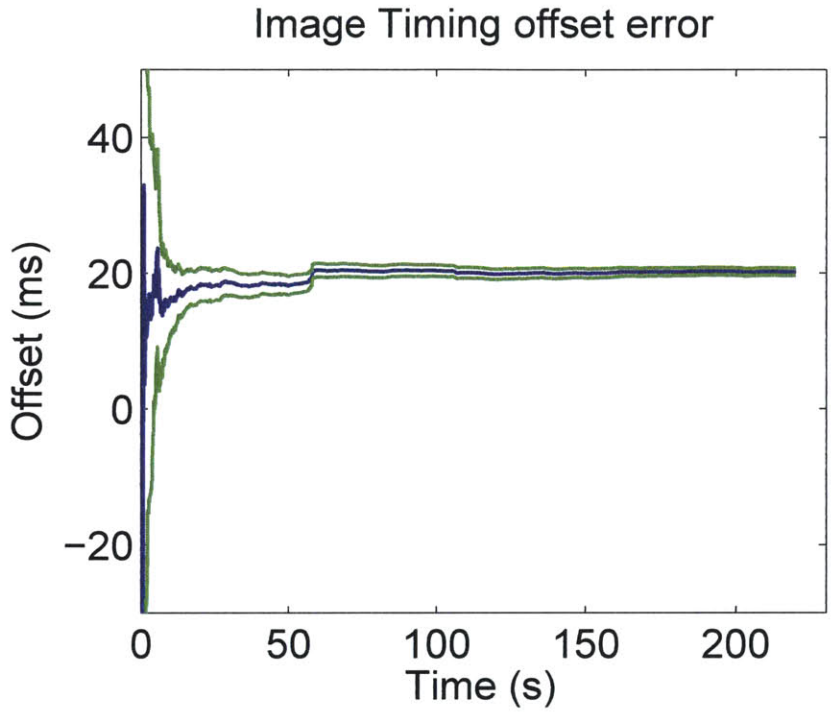


Figure 5-10: MSCKF-3D estimation of camera-IMU time delay during double loop walking trajectory. The blue line is the estimated timing offset and the green lines are the two-sigma error bounds.

Two additional datasets, shown in figures 5-11 and 5-12, were taken by walking the quadrotor though indoor office hallways. Since these hallways did not have AprilTags to provide truth data, the estimated trajectories are aligned with the outlines of the building hallways. From the hallway outlines, it is easy to see that the MSCKF-3D provides better trajectory estimates than the MSCKF. The other state estimates (i.e. vehicle orientation, IMU biases, camera-IMU transformation and timing offset) are similar to the state estimates from the double rectangular loop trajectory shown in figures 5-6 through 5-10.

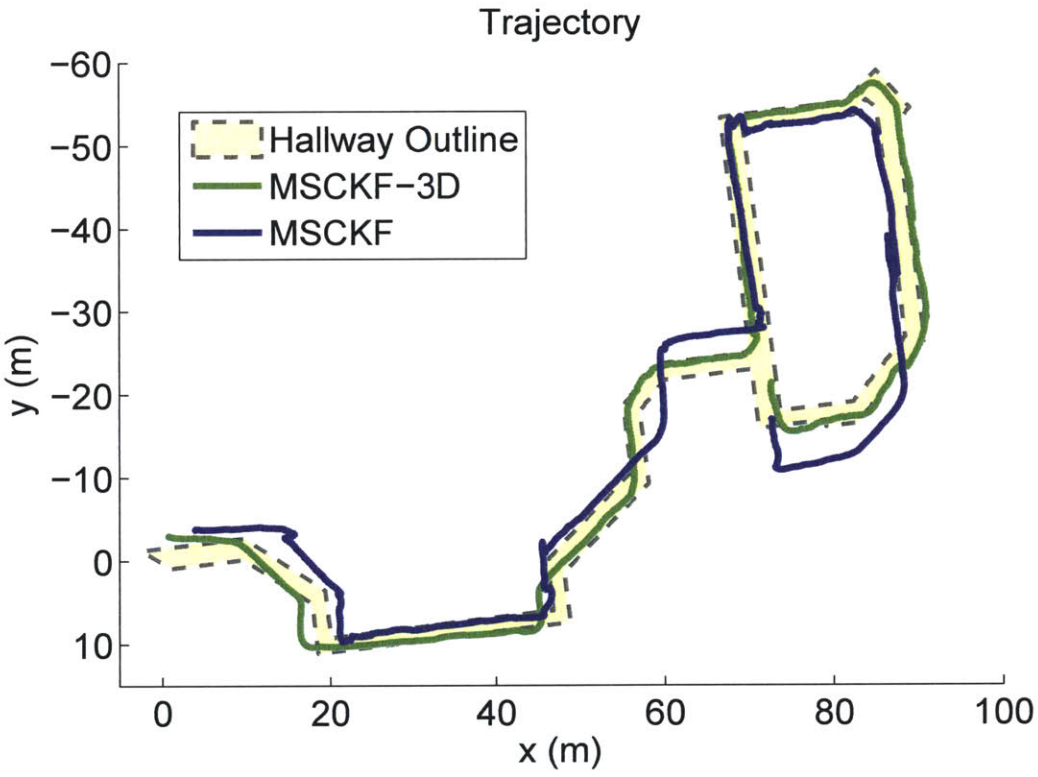


Figure 5-11: Trajectory obtained by walking the quadrotor around an office building. A long, windy path is taken and a loop is completed at the end. The estimated trajectories produced by the MSCKF and the MSCKF-3D (shown in blue and green respectively) are aligned with the hallways of the building (yellow).

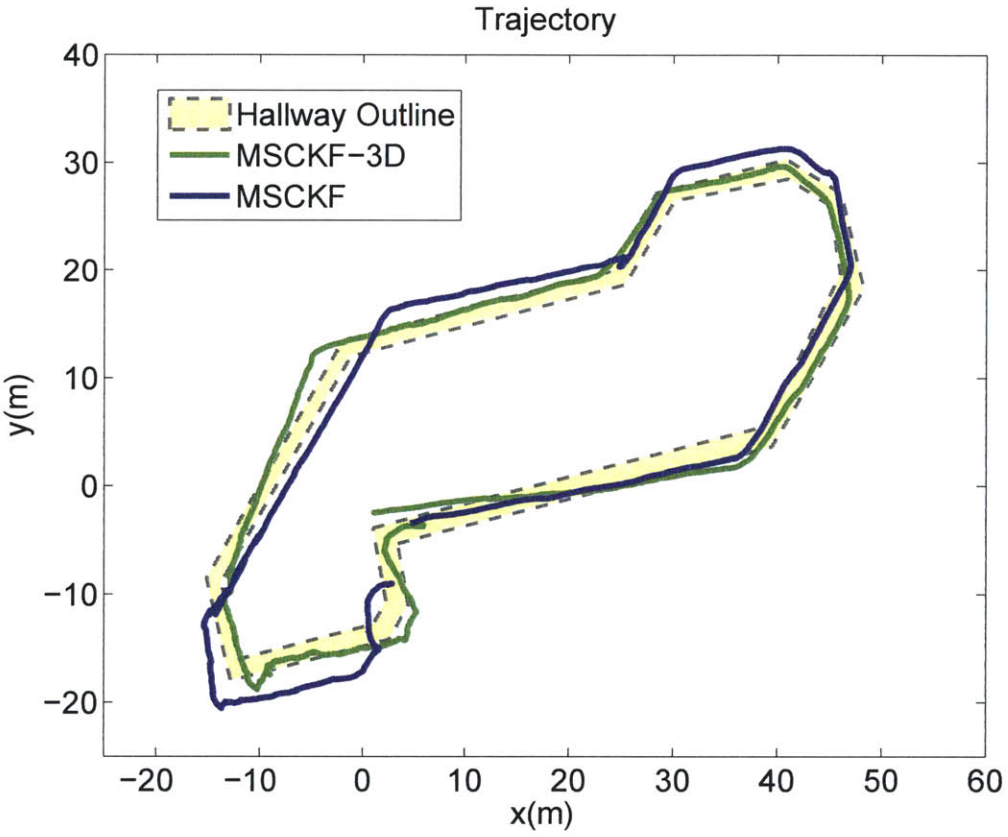


Figure 5-12: Trajectory obtained by walking the quadrotor around an office building. A windy loop is completed. The estimated trajectories produced by the MSCKF and the MSCKF-3D (shown in blue and green respectively) are aligned with the hallways of the building (yellow).

Figure 5-13 shows the trajectory estimated by the MSCKF-3D while the quadrotor was carried down two flights of stairs. In this dataset, the quadrotor started on the ground (which corresponds to a height of zero meters in the figure) and was picked up and held at about one meter above the ground. Then, the quadrotor was carried about four meters to the beginning of the staircase. Two flights of stairs were then descended.

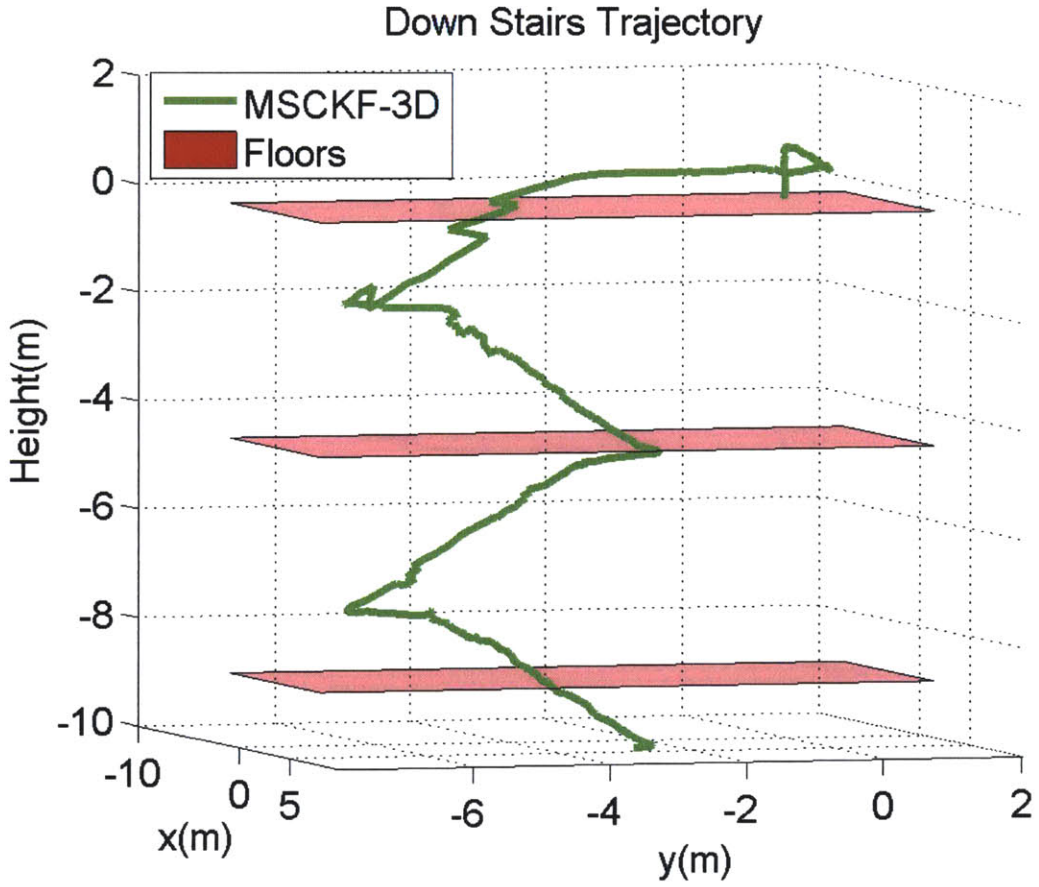


Figure 5-13: Trajectory obtained by walking down two flights of stairs. The estimated trajectory produced by the MSCKF-3D is shown in green. The red planes are the three levels of the building.

The MSCKF-3D overestimates how far down the quadrotor has traveled. Notice how the the trajectory ends about one meter below the bottom floor; since the quadrotor was being held, the true trajectory should end about one meter above the bottom floor. This is because the accelerometer in the IMU drifts as seen in figure 5-

8. Since the accelerometer is drifting in the same direction the quadrotor is traveling (i.e. downward), the total distance descended is slightly overestimated.

It should also be noted that the MSCKF filter was unsuccessful in estimating the trajectory of the staircase dataset. The staircase was dark and had very few features, making it very challenging to estimate the trajectory. The additional depth measurements enabled the MSCKF-3D to estimate the trajectory.

5.2.2 Flying Dataset

The next data set was taken by flying the quadrotor about ten meters to the end of a hallway, completing a 180 degree turn and flying the quadrotor back to its original position. Figure 5-14 shows the true trajectory of the quadrotor (from the AprilTags system) and the MSCKF-3D trajectory estimate and its corresponding error.

Figure 5-15 shows the position and velocity of the quadrotor as a function of time. In the estimate for the vertical position of the quadrotor, the negative z position corresponds to positive height above the ground. The MSCKF-3D accurately estimates the quadrotor lifting off, however it does not estimate the quadrotor returning to its position on the ground. This is, again, due to the bias in the accelerometer; the filter underestimates how much the quadrotor descends at the end of its flight.

The orientation of the quadrotor is shown in figure 5-16. The quadrotor is resting on the ground for the first 25 seconds of the dataset. As the quadrotor takes off for flight, notice the error bounds for the pitch and roll estimates clamp down on the estimates. Since there is more motion in the pith and roll axes, the filter is more accurately able to estimate the orientation of the quadrotor. The plot of quadrotor yaw clearly shows the quadrotor turning around in the hallway at about 60 seconds into the dataset and the returning in the opposite direction.

The IMU biases are estimated during the quadrotor flight and shown in figure 5-17. Since this dataset was short (only about 100 seconds) and the quadrotor flight begins at about 25 seconds, the accelerometer biases do not fully settle down as they did in figure 5-8.

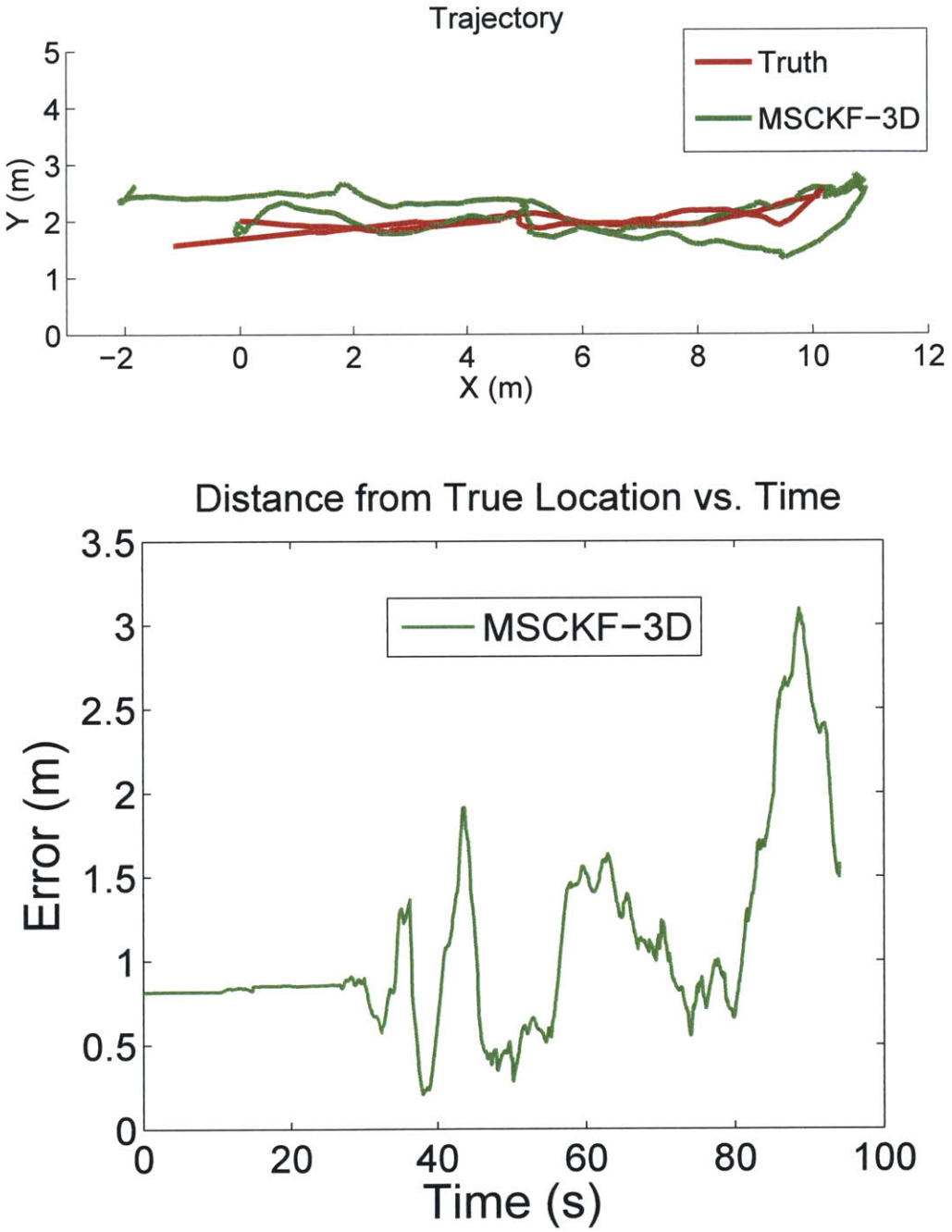


Figure 5-14: Trajectory obtained by flying the quadrotor in an office building. The quadrotor flies to the end of a hallway, turns 180 degrees and comes back to the starting position (shown in red). The estimated trajectory produced by the MSCKF-3D is shown in green. (Bottom) Distance between the true location and the estimated location by MSCKF-3D.

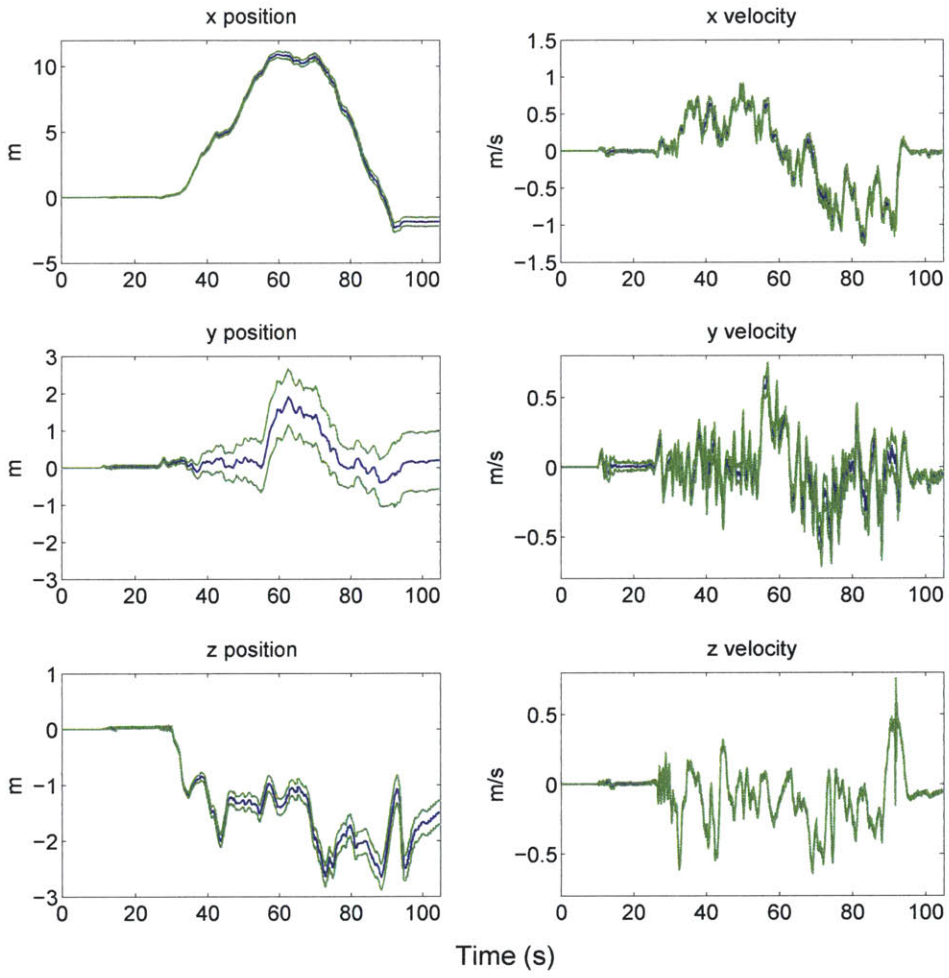


Figure 5-15: MSCKF-3D estimation of quadrotor position and velocity during flying trajectory. The blue lines are the estimates and the green lines are the two-sigma error bounds.

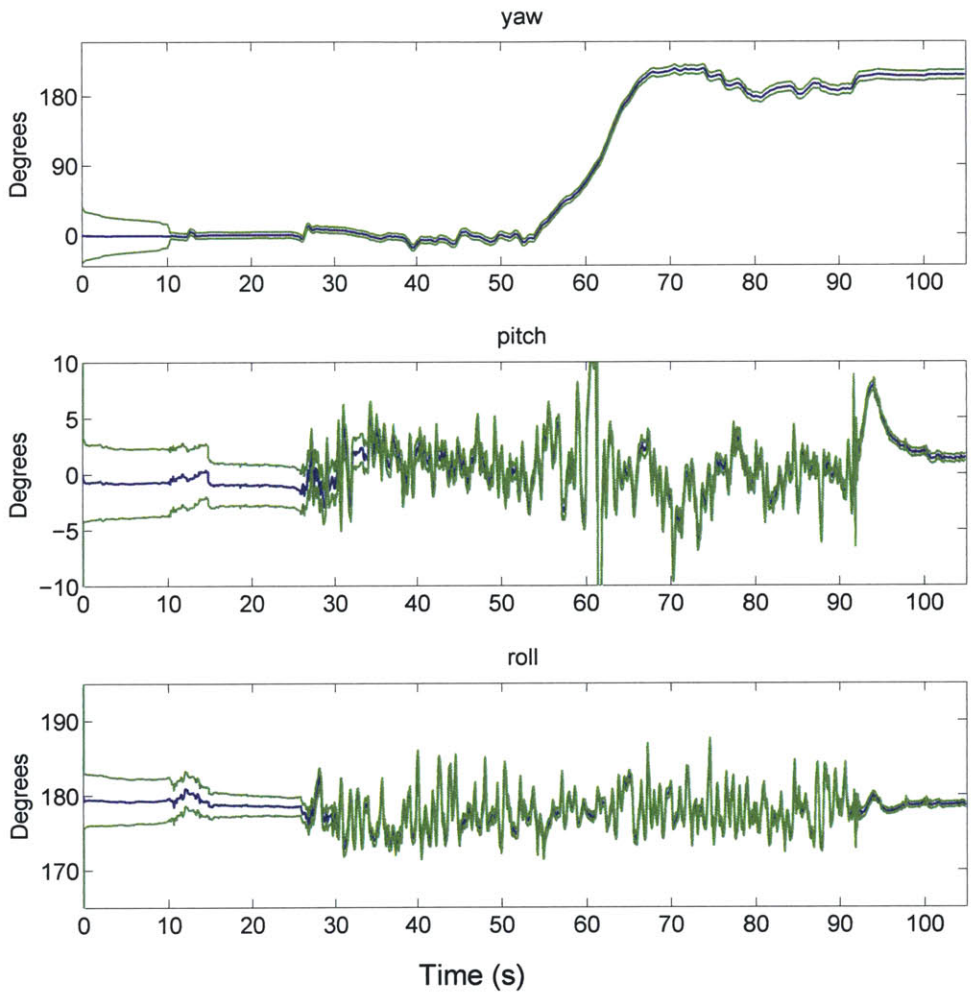


Figure 5-16: MSCKF-3D estimation of quadrotor orientation during flying trajectory. The blue lines are the estimates and the green lines are the two-sigma error bounds.

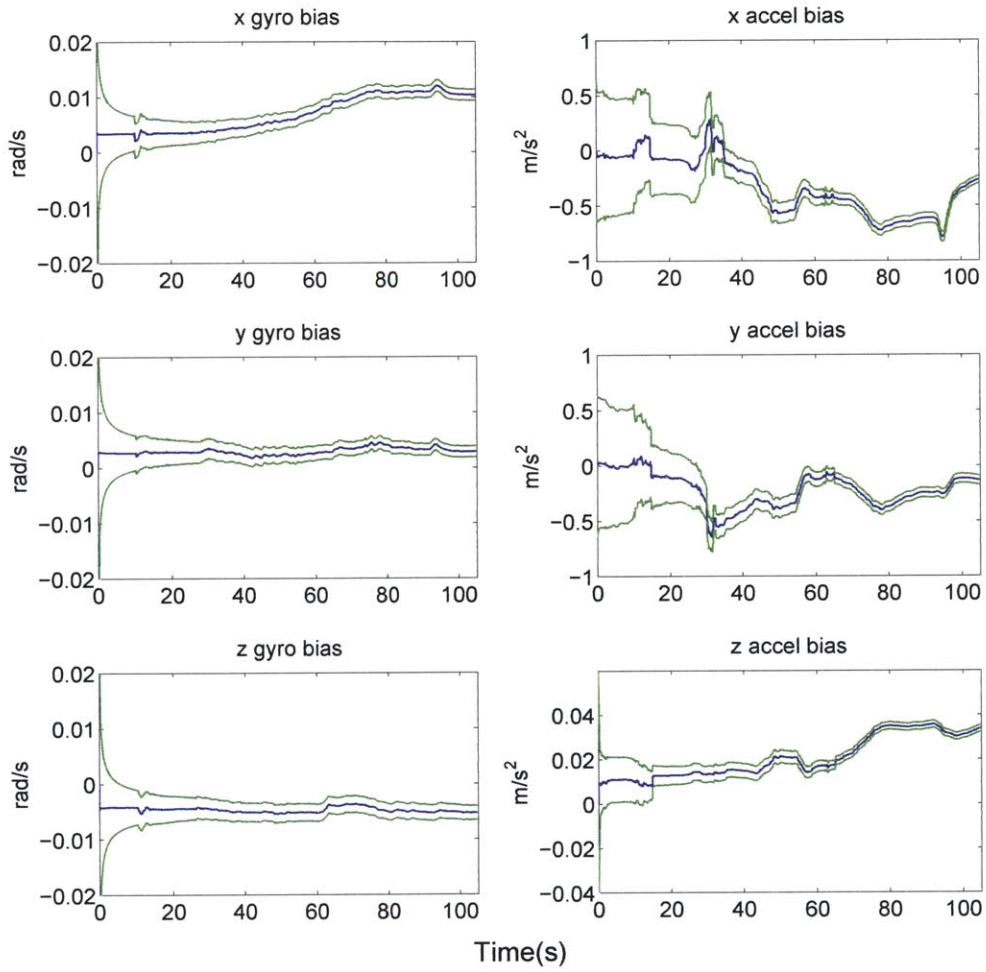


Figure 5-17: MSCKF-3D estimation of IMU biases during flying trajectory. The blue lines are the estimated biases and the green lines are the two-sigma error bounds.

Finally, as shown in figure 5-18, the MSCKF-3D estimates an approximately 20 ms timing offset between the timstamps on the IMU measurements and the camera measurements. This agrees with the timing offset predicted in other datasets.

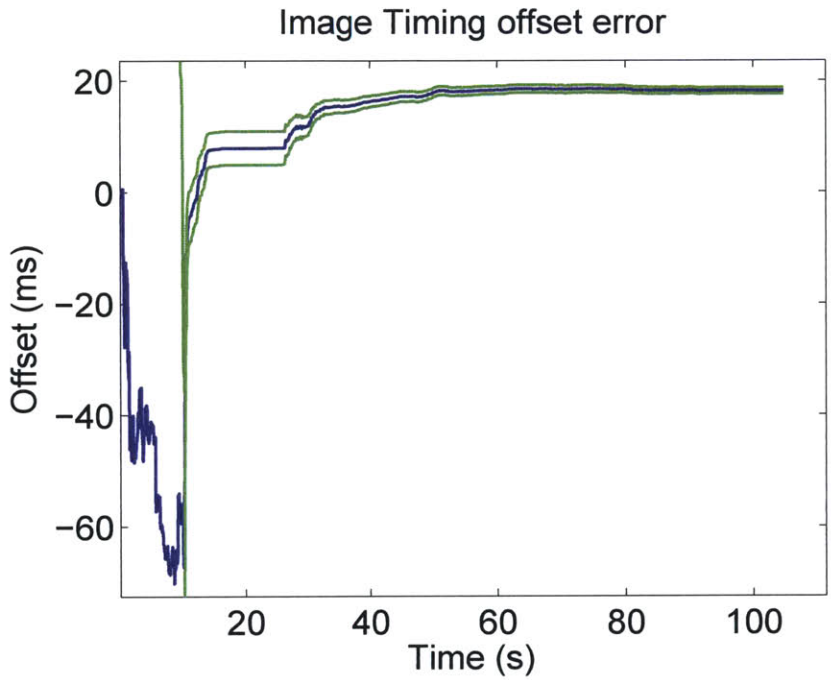


Figure 5-18: MSCKF-3D estimation of camera-IMU time delay during flying trajectory. The blue line is the estimated timing offset and the green lines are the two-sigma error bounds.

It is noted that the MSCKF filter was unsuccessful in estimating the trajectory of the quadrotor during this flight. As opposed to datasets where the quadrotor was carried, this flying dataset is more challenging because of the vibrations and jitter of the quadrotor during flight. This directly affects the vision measurements because the features are more difficult to track from frame to frame since the quadrotor, and therefore the onboard camera, is jittering and creating image blur. The additional measurements provided by the depth sensor enabled the MSCKF-3D to use more information from the few vision measurements to estimate trajectory.

5.3 Summary of MSCKF and MSCKF-3D

Accuracy

The MSCKF-3D estimates vehicle trajectory more accurately than does the MSCKF. Table 5.2 summarizes the position errors from both filters for the three data sets with available truth data. Even though the other datasets do not have truth data available, it is still easily seen, by comparing the trajectory estimates with the office floorplans that the MSCKF-3D provides better trajectory estimates than does the MSCKF. Additionally, only the MSCKF-3D was able to estimate the vehicle trajectory in the staircase dataset and the flying dataset.

Table 5.2: Accuracy of the MSCKF and the MSCKF-3D

Data Set	Trajectory length (m)	Filter	Position RMSE (m)	Position RMSE (% Total Trajectory Length)
Single Loop-Walking	99.01	MSCKF	4.21	4.31
		MSCKF-3D	1.99	2.01
Double Loop-Walking	198.98	MSCKF	16.88	8.48
		MSCKF-3D	3.48	1.75
Flying	25.65	MSCKF	-	-
		MSCKF-3D	1.22	4.76

Computational Comparison

Each time a new image is recorded, the filters perform a vision update as described in sections 3.2.4 and 4.3. Since the MSCKF-3D processes depth data in addition to the RGB data, each vision update takes longer in the MSCKF-3D than in the MSCKF. This is driven by the matrix inversions performed in the weighted least squares estimation of feature location, described in section 4.3. Table 5.3 compares the average vision update times for the MSCKF and the MSCKF-3D in various data sets. In summary, the MSCKF-3D takes about 15% longer than the MSCKF per vision update.

Table 5.3: Computation time per vision update for various datasets

Data Set	Filter	Average time per vision update (ms)	% Longer than MSCKF vision updates
Single Rectangular Loop	MSCKF	94.9	-
	MSCKF-3D	104.4	10.1
Double Rectangular Loop	MSCKF	55.6	-
	MSCKF-3D	68.5	23.2
Windy Path with end loop (figure 5-11)	MSCKF	83	-
	MSCKF-3D	97.1	17.0
Z-shaped loop (figure 5-12)	MSCKF	86.3	-
	MSCKF-3D	99.3	15.1

Chapter 6

Conclusions

A visual inertial odometry algorithm for a system consisting of an IMU, a monocular camera and a depth sensor is presented in this thesis. The estimation algorithm, termed MSCKF-3D, extends the multi-state constraint Kalman filter (MSCKF), proposed by Mourikis and Li [34, 29], to incorporate depth measurements, which enable the system to produce three-dimensional feature observations rather than two-dimensional observations. Each feature observation is approximated as a multivariate Gaussian distribution, as proposed by Dryanovski et al. [11]. Then the global feature location is estimated using a weighted least squares estimate with all the feature observations. The MSCKF-3D also estimates the timing offset between the camera and the IMU measurements.

The MSCKF-3D algorithm is compared to the original MSCKF algorithm using real-world data obtained by flying a custom-built quadrotor in an indoor office environment. Both the MSCKF and the MSCKF-3D used the same vision and inertial measurements. The MSCKF-3D, however, also used measurements obtained from a depth sensor. Processing the depth measurements resulted in the MSCKF-3D taking slightly longer to perform vision updates. As demonstrated in the experimental results, the MSCKF-3D consistently outperformed the MSCKF in its estimation of the vehicle trajectory.

6.1 Future Work

The performance of the MSCKF-3D can be further improved by integrating measurements from additional sensors into the estimation algorithm. The custom-built quadrotor is also equipped with an optic flow sensor and a barometer. The next step is to include measurements from these sensors into the estimator.

Currently, the MSCKF-3D post-processes the vision and inertial data; the next goal is to run the filter on board the quadrotor for real-time state estimation during flight. This entails coding the MSCKF-3D filter in C and integrating it into the ROS framework, which is the onboard computing and operating framework for the quadrotor. Using the MSCKF-3D for state estimation is one step toward achieving a fully autonomous quadrotor capable of localization and navigation of unknown environments.

Bibliography

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2008.

[2] *Asus Xtion Pro Specification*, Asus. [Online]. Available: http://www.asus.com/Multimedia/Xtion_PRO_LIVE/specifications.

[3] Tim Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *Robotics Automation Magazine, IEEE*, 13(3):108–117, Sept 2006.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[5] Roland Brockers, Sara Susca, David Zhu, and Larry Matthies. Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs. volume 8387, pages 83870Q–83870Q–10, 2012.

[6] Averil B. Chatfield. *Fundamentals of High Accuracy Inertial Navigation.* 1997.

[7] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of uav. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2010–2016, April 2007.

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

[9] D.D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 221–228, Jan 2005.

[10] Tue-Cuong Dong-Si and A.I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5655–5662, May 2011.

[11] I. Dryanovski, R.G. Valenti, and Jizhong Xiao. Fast visual odometry and mapping from rgb-d data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2305–2310, May 2013.

[12] H. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics Automation Magazine, IEEE*, 13(2):99–110, June 2006.

[13] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2815–2821, Oct 2012.

[14] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90, June 2012.

[15] Arthur Gelb, Joseph F. Kasper, Raymond A. Nash, Charles F. Price, and Arthur A. Sutherland, editors. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[16] *Gigabyte brix GB-XM1-357.* [Online]. Available: http://www.gigabyte.com/products/product-page.aspx?pid=4581# sp.

[17] Christopher G. Harris and J. M. Pike. 3d positional integration from image sequences. *Image Vision Comput.*, 6(2):87–90, 1988.

[18] Guoquan P. Huang, A.I. Mourikis, and S.I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 473–479, May 2008.

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

[20] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[21] G. Klein and D. 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, Nov 2007.

[22] M. Kleinert and S. Schleith. Inertial aided monocular slam for gps-denied navigation. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 20–25, Sept 2010.

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

[24] Kurt Konolige, Motilal Agrawal, and Joan Sol. Large scale visual odometry for rough terrain. In *In Proc. International Symposium on Robotics Research*, 2007.

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

[26] M. Li and A. I. Mourikis. Consistency of ekf-based visual-inertial odometry. Technical report, University of California, Riverside, Dept. of Electrical Engineering, 2011.

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

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

[29] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visualinertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[30] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

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

[32] J. Ma, S. Susca, M. Bajracharya, L. Matthies, M. Malchano, and D. Wooden. Robust multi-sensor, day/night 6-dof pose estimation for a dynamic legged vehicle in gps-denied environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 619–626, May 2012.

[33] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University doctoral dissertation, Stanford University*, number CMU-RI-TR-80-03. September 1980.

[34] A.I. Mourikis and S.I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572, April 2007.

[35] *MPU-6000 Product Specification*, Invensense. [Online]. Available: http://invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf.

[36] D. Nister, O. Naroditsky, and J. 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–I–659 Vol.1, June 2004.

[37] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[38] P. Pinies, T. Lupton, S. Sukkarieh, and J.D. Tardos. Inertial aiding of inverse depth slam using a monocular camera. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2797–2802, April 2007.

[39] *PX4FMU Autopilot / Flight Management Unit.* [Online]. Available: http://pixhawk.org/modules/px4fmu.

[40] *Robot Operating Software (ROS).* [Online]. Available: http://www.ros.org/.

[41] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer, 2006.

[42] S.I. Roumeliotis, A.E. Johnson, and J.F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 4, pages 4326–4333 vol.4, 2002.

[43] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92, Dec 2011.

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

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

[46] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Proceedings of the Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 267–288, Corvallis, Oregon, 1986. AUAI Press.

[47] H. Strasdat, J. M M Montiel, and A.J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664, May 2010.

[48] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* 2001.

[49] S. Weiss, M.W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964, May 2012.

[50] B. Williams, N. Hudson, B. Tweddle, R. Brockers, and L. Matthies. Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 431–438, May 2011.