AUTONOMOUS VISUAL-INERTIAL NAVIGATION AND ABSOLUTE VISUAL SCALE ESTIMATION

BY

XINKE DENG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Timothy Bretl

ABSTRACT

In this thesis, we present a system that uses a single camera and an inertial measurement unit (IMU) to navigate an Unmanned Aerial Vehicle (UAV) in a previously unknown environment. The approach consists of two parts. First, we apply a state-of-the-art simultaneous localization and mapping (SLAM) method to the video stream of a onboard camera. From the SLAM system, an up-to-a-scale pose of the camera is estimated, because the absolute size of the environment cannot be estimated with a single camera. Second, the estimated pose is fused with the data from IMU to resolve the scale ambiguity.

While analyzing the performance of the system, we find that the convergence rate of scale decreases when the magnitude of scale increases. This relationship has not been demonstrated and explained before. In this thesis, we present an analysis and explanation of this phenomenon.

To my family for their love and support.

ACKNOWLEDGMENTS

This thesis would not have been possible without the help of many people. First, I would like to thank my adviser, Professor Timothy Bretl for all his encouraging support and insights. As an international student, confronting the language barrier and making a clear presentation are challenging. Professor Bretl made lots of efforts to help me overcome these challenges. Conversation with Professor Bretl also made me think about problems more critically and carefully. From him, I learned that asking a good question is more important than solving one. In addition, I would like to thank all the members of the Bretl Research Group during my time as a Master's student (Navid Aghasadeghi, Aadeel Akthar, Andy Borum, Kevin Chen, Joe DeGol, David Hanley, Joe Nance, Jessica Mullins, Mary Nguyen, Jamie Norton, Jacob Wagner, Sean Yen, Kazuaki Iida, and Kyung Yun Choi). Each of you make the lab a great place to work. In particular, I would like to thank Joe DeGol and David Hanley for their many hours of help and discussion. Finally, I would like to thank my family. Their love and support has enabled me to write this thesis. This work is supported in part by the National Science Foundation Grant No. 14-46765 and 14-27111.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Recently, Unmanned Aerial Vehicles (UAV) have been a popular research topic not only in the defense industry but also for civilian applications. Thanks to their agility and relatively low cost, UAVs can be applied in various tasks such as aerial photography, surveillance, and search and rescue. For example, UAVs were used in the 2010 Haiti earthquake's rescue mission to count the number of tents and identify the aid requirement [1]. Amazon is working on the idea of using a UAV to deliver its commodities. With the data collection equipment, UAVs can fly around a construction site and help managers keep track of construction progress [2].

Multirotor UAVs, in comparison to the traditional fixed wing UAVs, are of special interest because of their ability to perform fast maneuvers and hover [3] [4]. In addition, the ability to vertically take off and land enables multirotor UAVs to maneuver in a constrained, indoor environment.

Autonomous mobile robots need to navigate with a wide range of sensors. Figure 1.1 shows a classification of sensors used in robotic navigation [5]. Ideally, we want a sensor for navigation to have high update rate (framerate) but low drift both spatially and temporally. Inertial measurement units (IMUs) have, typically, a framerate on the order of 1KHz, but the accumulation of error makes the estimation of position drift very quickly. Camera and laser-scanner based odometry approaches provide measurements at medium rates and do not drift as long as the vehicle maintains its current position [6]. The Global Positioning System (GPS) and Laser Trackers can provide drift free position estimation, but the framerate is low.

Currently the vast majority of UAVs navigate in their environments using GPS as a reference for position. Unfortunately, GPS signals are often unreliable or non-existent in environments where UAVs are operating such as urban canyons, indoors, and any areas where direct satellite line-of-sight is impeded.

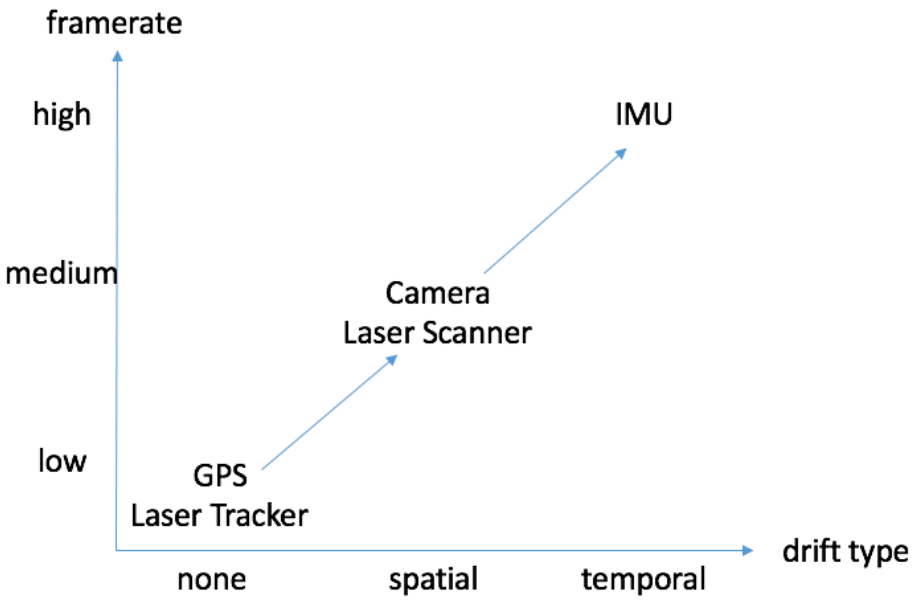

Figure 1.1: Classification of sensors in robotic navigation [5]

Sensor Example	Price	Weight	Framerate
MV Bluefox-MLC 200 WG	310 USD	16 gram	up to 90 Hz
HOKUYO UST-20LX	2858 USD	130 gram	up to 40 Hz

Table 1.1: Sensor comparison

Since GPS navigation solutions are not always reliable, UAVs require other independent navigation systems. In comparison to laser scanners, cameras are usually of lower price, are lighter, and have faster framerates. Using the MV Bluefox-MLC 200 WG camera and HOKUYO UST-20LX laser scanner as examples, the price, weight, and framerate are listed in the table 1.1. These properties make camera a popular sensor for robot navigation.

In the context of camera based robotic navigation, some research focus on using fiducial markers distributed along a predetermined path [7] [8] to resolve the localization issue. The downsides with these approaches is that they require additional equipment, time, and prior access to the target location to set up.

Another alternative to the markers-based navigation is vision-based simultaneous localization and mapping (VSLAM) which has recently become possible with the steady advances in computing power. The general idea of Visual SLAM is: based on the images collected by a camera, a map is

established, then the camera can be localized based on the map. The map can be used to avoid the obstacles, and gain more understanding about the environment; the estimated position can be used to stabilize the robot or to make the robot follow a certain path.

One of the most important problems of monocular VSLAM is the ambiguity in the scale, which means the camera cannot give the absolute size of the environment (e.g. size in SI units). This is called scale ambiguity. To resolve this ambiguity, IMU data is combined with the output of VSLAM, which is called visual inertial sensor fusion.

While analyzing the performance of scale estimation, we find that the convergence rate of scale decreases when the magnitude of scale increases. This phenomenon has not been presented before.

1.1 Outline

In Chapter 2, we introduce one of the state-of-the-art VSLAM algorithms: PTAM, which is short for Parallel Tracking and Mapping [9]. The visual inertial sensor fusion methods are presented in Chapter 3. In Chapter 4, we present the relationship between the scale convergence rate and the magnitude of scale. In Chapter 5, we summarize this thesis and give an outlook of future research.

CHAPTER 2

SIMULTANEOUS LOCALIZATION AND MAPPING

Simultaneous localization and mapping (SLAM) systems are crucial for autonomous robot navigation in a previously unknown environment. In this chapter, we provide background information about SLAM.

The basic idea of SLAM is that a map of the surrounding environment is continuously updated based on the robot's onboard sensor data (mapping), while simultaneously the position and orientation of the robot is estimated based on the built map (tracking). In the area of vision-based SLAM (VSLAM), people usually use the position of landmarks in an absolute global frame (also called world frame) to represent a map of the environment. The landmarks are typically distinctive feature points and patches, which are also referred to as keypoints. The position and orientation of the camera can then be estimated based on the estimated position of keypoints. When the camera enters a new environment or identifies new landmarks, new keypoints are added into the map, and the map is updated.

Although the basic idea of VSLAM seems simple, actually solving the VSLAM problem is very challenging. First of all, in order to run the VSLAM algorithm onboard, the computational complexity has to be considered: especially when the map keeps growing. In addition, rapid acceleration and erratic motion of a camera may cause the failure in tracking the landmarks. How to make the VSLAM systems more robust to these motion is still an open challenge. Moreover, monocular based VSLAM systems suffer from the scale ambiguity. While the metric information is not very important in the structure-from-motion context, it is crucial in robotic control. Apart from the above problems, robustly dealing with dynamic environments and efficiently detecting loop-closure are also very popular research areas.

Currently, the VSLAM problem is addressed in different ways. Filtering based methods marginalize out past poses and summarize the information gained over time with a probability distribution. Keyframe based methods

retain the optimization approach, but computationally must select only a small number of past frames to process [10]. On the side of filter based VSLAM, two successful systems are [11] and [12]. The first one is based on the extend Kalman filter (EKF) and the map is represented as states. The second one adapts the FastSLAM 2.0 presented in [13] and uses particle filter to estimate the map and motion of the camera. In the filter based VSLAM systems mentioned above, tracking is tightly coupled with mapping and every camera frame is used for both tracking and mapping.

The focus of this chapter: Parallel Tracking and Mapping (PTAM) which was proposed in [9] is a keyframe based VSLAM algorithm. In PTAM, keyframes refer to the subset of camera images which are used to build the map and the corresponding position and orientation of the camera when these images are taken. Each keyframe also stores a four-level image pyramid: level 0 corresponds to the original image and level 3 corresponds to the most downsampled image. In [10], the authors show both theoretically and experimentally that keyframe bundle adjustment based VSLAM outperforms filtering based ones since it gives the most accuracy per unit of computing time.

The structure of this chapter is as follows: first, the basic algorithm will be shown in the section 2.1. Then the three main components: initialization, tracking and mapping will be further discussed in section 2.2 to 2.4. Originally PTAM was designed for small augmented reality workspace. However, this is not always the case for aerial robotic missions. In section 2.5, the adaption for large scale environment by Weiss in [14] is discussed.

2.1 Overview

The PTAM flow chart is shown in figure 2.1. Different from common VSLAM algorithms with the frame-to-frame characteristic, such as [11], PTAM splits the VSLAM task into two separate threads: the tracking thread and the mapping thread. Since both of the threads need a map to work on, a map is initialized using the stereo technique (5 point algorithm) shown in [15]. Once the initial map is built, the tracking thread will run to estimate the position and orientation of the camera relative to the world frame. When the displacement of the camera exceeds certain threshold, new keyframes will be

added to update the map.

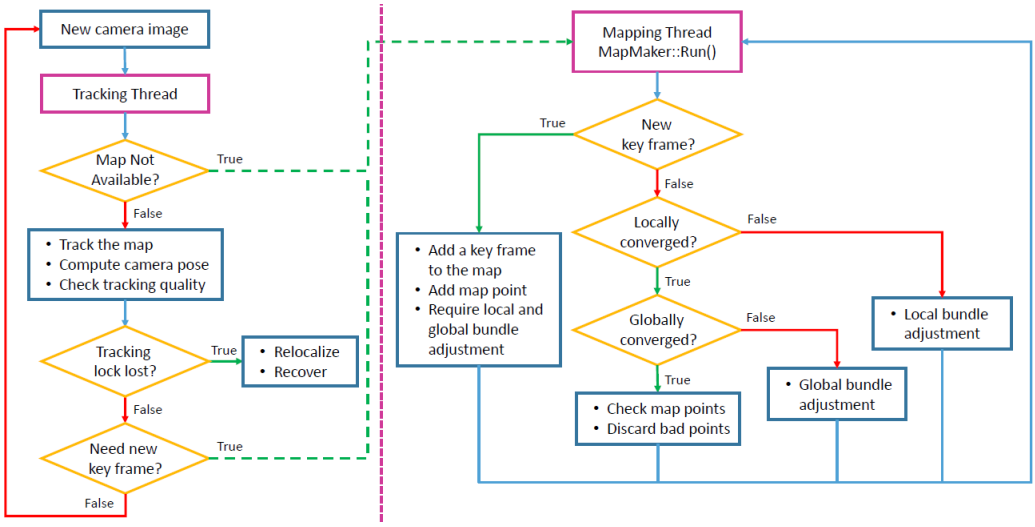

Figure 2.1: PTAM flow chart

2.1.1 Notation

The notation conventions are as follows:

- All the matrices are denoted by upper-case, bold letters and vectors are denoted by lower-case, bold letters.
- Number 0 represents the world frame, character "C" represents the camera frame and "C_j" represents the camera frame for j-th keyframe.
- The coordinate transformation from the world frame to camera frame is denoted as $\mathbf{T}_0^C \in SE(3)$ and the coordinate transformation from the camera frame for $j1$-st keyframe to $j2$-nd keyframe is denoted as $\mathbf{T}_{C_{j1}}^{C_{j2}} \in SE(3)$.
- The i-th keypoint is represented as $\mathbf{p}_i$ and it is represented in the world frame in the homogeneous form, i.e. $\mathbf{p}_i \doteq \mathbf{p}_i^0 = [x_i^0, y_i^0, z_i^0, 1]^\mathbf{T}$.
- The projective coordinate of the i-th keypoint in the current image is represented as $\begin{bmatrix} u_i \\ v_i \end{bmatrix} = CamProj(\mathbf{T}_0^C \mathbf{p}_i)$ and $CamProj$ represents the pin-hole camera projection model.

- The measured coordinate of the i-th keypoint in the current image is represented as $\hat{\mathbf{p}}_\mathbf{i} = \begin{bmatrix} \hat{u}_i \\ \hat{v}_i \end{bmatrix}$.
- The reprojection error is defined as $e_i = \hat{\mathbf{p}}_\mathbf{i} - CamProj(\mathbf{T}_0^C \mathbf{p}_i)$.

2.2 Initialization

The necessity of initialization comes from the "chicken-or-egg" nature of the keyframe based VSLAM problem: tracking needs a map to triangulate the camera's motion while mapping needs the position and orientation to locate the keypoints. This problem is addressed in PTAM by incorporating user cooperation. The steps are:

1. The user sends out a command to take the first keyframe. Keypoints in the first keyframe are detected by the FAST corner detector [16].
2. The user smoothly translates (keeping the rotation small) the camera some small distance. When the camera is moving, the keypoints are tracked. This is shown as figure 2.2.
3. When the distance camera moves is enough, the user sends out the command to take the second keyframe. At this time, the correspondences between the tracked keypoints are used by the 5 point algorithm and RANSAC to estimate the essential matrix [17] and build the basic map.
4. The basic map is refined through bundle adjustment.

Note that the ambiguity of scale is first shown in the initialization of PTAM. When applying the 5 point algorithm to determine the essential matrix, only 5 out of the 6 degrees of freedom can be determined. In order words, the essential matrix can only be determined up to a scale, leaving translation $\mathbf{t}$ undetermined. In the real implementation, the translation $\mathbf{t}$ is assumed as a constant number.

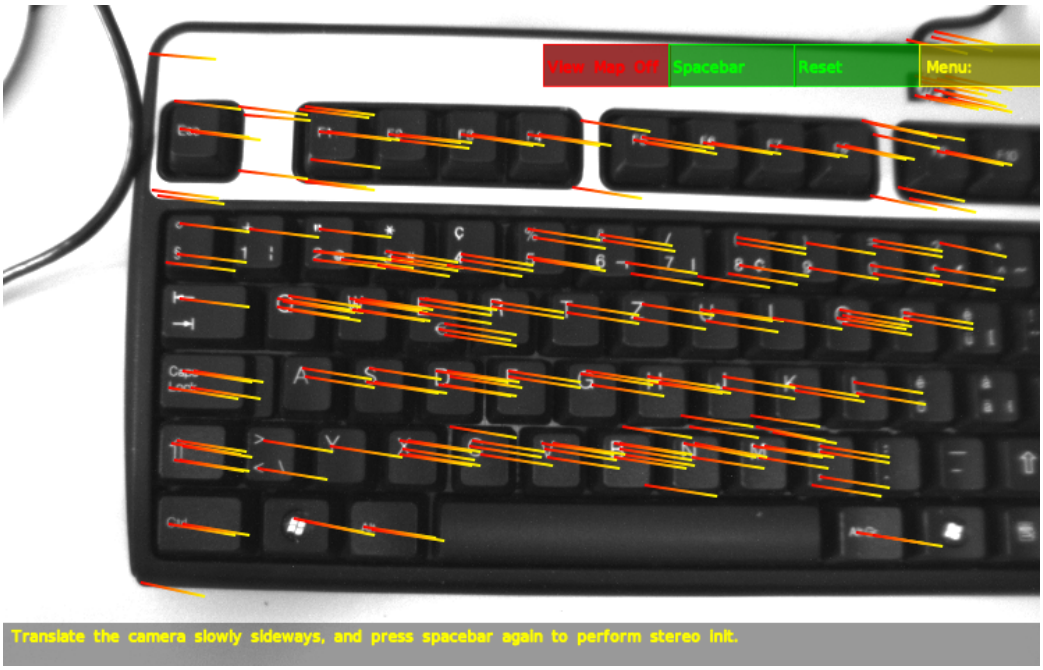

Figure 2.2: Tracking of the keypoints in the first keyframe

2.3 Tracking

The tracking thread compares the features in the images with the already existing map and thereby estimates the position and orientation (pose) of the camera. First, the pose of the camera is estimated by applying a simple motion model such as a decaying velocity model. Then data association is performed by projecting the keypoints into the camera frame. After the data association is done, the algorithm computes the pose of the camera by minimizing the projection error.

$$C^* = \underset{C}{argmin} \sum_{i=1}^{N} Obj\left(\frac{|e_i(\mathbf{p_i}, \hat{\mathbf{p}}_\mathbf{i}, C)|}{\sigma_i} + \sigma_T\right) \tag{2.1}$$

According to [9], $Obj(\cdot, \sigma_T)$ is the Tukey biweight objective function mentioned in [18] and σ_T is a robust estimate of the distribution's standard deviation derived from all the reprojection errors.

2.4 Mapping

2.4.1 Adding Keyframes

In order to achieve real time computation and a more accurate map, a subset of camera images (i.e. keyframes) are used to build the map of the environment. A camera image is selected as a keyframe when the following criterion are satisfied:

- Tracking quality is good.
- Mapping does not exceed the computational limit.
- The camera must be more than a minimum distance from the previously generated keyframe positions. One of the following criteria should be satisfied: 1)Median difference between observation and projection should be larger than a threshold; 2)Normalized distance between current position and last keyframe generated position should be greater than a threshold.

Figure 2.3 illustrates the generation of new key-frames, the red vertical lines represent the generation of a new keyframe. From the figure, we can see either the reprojection distance or the normalized spatial distance exceeding certain thresholds (the green line) causes the generation of a new keyframe.

2.4.2 Bundle Adjustment

In the keyframe based VSLAM, bundle adjustment can be defined as refining the keypoints' position $\mathbf{p}_i$ and the camera position and orientation for keyframe C_j simultaneously [9]. This is done by minimizing the robust objective function

$$\{\{C_2 \cdots C_K\}, \{\mathbf{p}_1 \cdots \mathbf{p}_N\}\} = \underset{C_2 \cdots C_K, \mathbf{p}_1 \cdots \mathbf{p}_N}{argmin} \sum_{j=1}^{K} \sum_{i=1}^{N} Obj(\frac{\left|e_i^j\right|}{\sigma_i^j} + \sigma_T). \quad (2.2)$$

This is called global bundle adjustment because it refines all the keypoints and keyframes simultaneously. For some computation critical situations, global bundle adjustment is not feasible. Only part of keyframes and corresponding keypoints are refined. This is called local bundle adjustment.

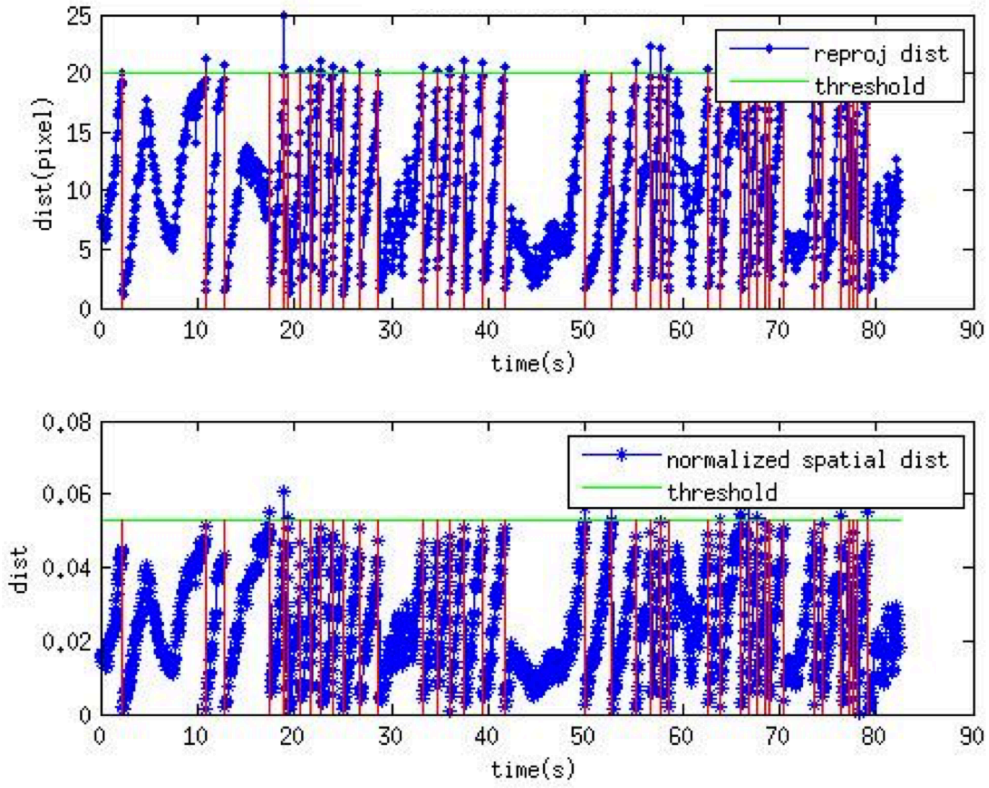

Figure 2.3: Generation of new keyframes. Red lines represent the generation of new keyframes.

2.5 Adaptation for Large Scale Environment

The PTAM software package[1] used in this thesis is implemented by the researchers from ETH Zurich Autonomous Systems Lab[2]. In order to adjust PTAM to a large scale environment, the algorithm is modified to decrease the computational complexity. The adaptations are:

- The modified PTAM algorithm only keeps local map (only store N closest keyframes which ensures constant complexity).
- The modified PTAM algorithm discards original image for map feature and just uses some more salient features to do the mapping. By doing this, the number of mapping outliers will be reduced significantly.
- In tracking, FAST corners are replaced by AGAST corners to get better performance.

Figure 2.4 shows the PTAM GUI. The pose of the camera for which keyframe is shown as a coordinate frame along with a point-cloud based map of the scene.

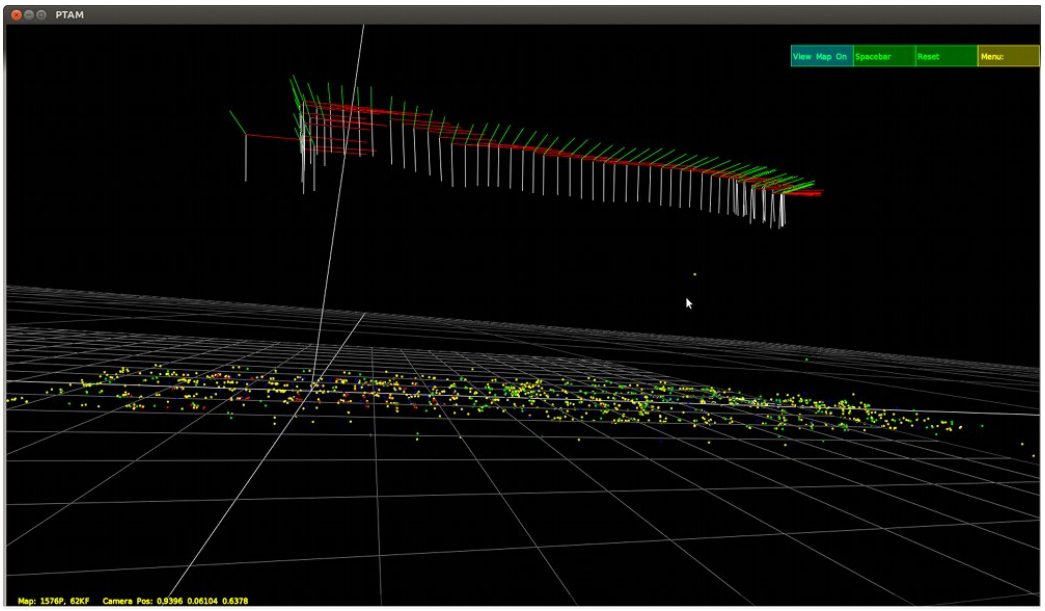

Figure 2.4: PTAM GUI

[1] http://wiki.ros.org/ethzasl_ptam
[2] http://www.asl.ethz.ch/

CHAPTER 3

VISUAL INERTIAL SENSOR FUSION

As discussed in the Chapter 2, PTAM suffers from the scale ambiguity. In order to resolve this ambiguity, information from other sensors which contain metric information is fused with the information from the camera. For example, in [19], the authors use an ultrasound altimeter to estimate the scale. Ultrasound sensors, however, cannot provide reliable distance measurements when the sensor reading is beyond a certain range. This is also the problem for other range based sensor such as stereo rigs. Inertial measurement units (IMUs) are commonly equipped on aerial robots. In addition, IMU can provide directly the vertical direction (i.e. the direction of gravity in the body frame) [20] [21] [22] and metric information to estimate the scale [23]. IMUs act as a good complement to a camera.

In this chapter, the methods of fusing visual and inertial data are discussed. To begin with, a brief review of inertial sensing is presented. Then, two types of visual inertial methods are introduced and compared. In this thesis, we will focus on the loosely-coupled visual inertial sensor fusion based on the extended Kalman filter (EKF) and the relative part is Section 3.3. We implemented the loosely-coupled visual inertial sensor fusion system described in [24], and the experimental results are shown in Section 3.4.

3.1 Inertial Measurement Unit

Inertial measurement units (IMUs) are sensors which exploit inertia to measure the linear motion (specific force) and angular motion (angular rate). Usually, an IMU consists of three orthogonal accelerometers to measure the specific forces and three orthogonal gyroscopes to measure the angular rate. Figure 3.1 shows the arrangement of accelerometers and gyroscopes of an orthogonally arranged IMU. Usually, the IMUs on aerial robots are strap-

down IMUs which means the IMUs are fixed in the body frame (the x^b, y^b and z^b in figure 3.1 represent the axes in the body frame). Figure 3.2 shows the microcontroller containing strap-down IMU of the UAVs from Ascending Technologies.

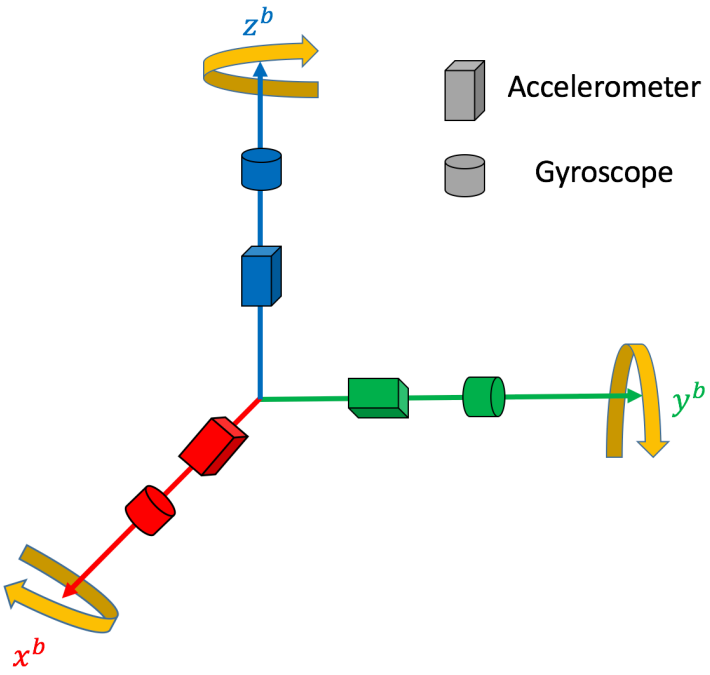

Figure 3.1: Components in an orthogonally arranged IMU

An IMU uses its accelerometers to measure the specific force. The physical principle of a single-axis accelerometer is shown as figure 3.3 [25]. The dynamics of the system can be expressed as:

$$\ddot{x}(t) + 2\zeta\omega_n\dot{x}(t) + \omega_n^2 x(t) = \ddot{y}(t) \tag{3.1}$$

Where ζ and ω_n represent the damping and natural frequencies of this system; they are determined by the spring constant k, the damper constant b, and the mass m. In this case, we can use the deviation of the mass from the equilibrium point (x) and its first and second derivatives to represent the acceleration of $\ddot{y}$, which represents the acceleration of the body frame. Note here the gravity will only change the equilibrium point of this mass-spring-damper system; therefore, it cannot be measured separately.

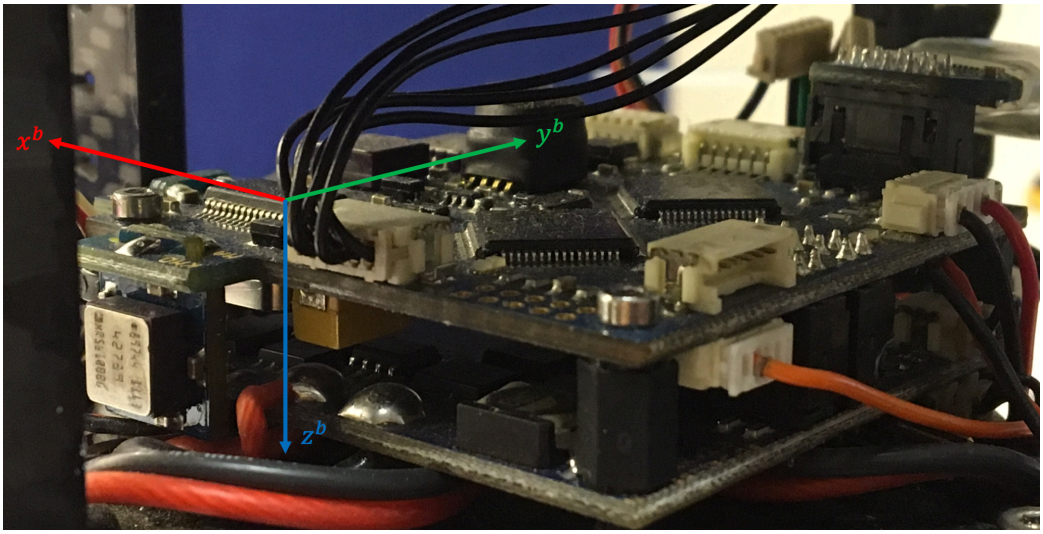

Figure 3.2: Microcontroller with a strap-down IMU on Asctec UAV

The measurement of angular rate is relative to measuring Coriolis force. When gyroscope spinning disk which has moment of inertia $\mathbf{I}$ rotates with angular velocity $\mathbf{p}$, if the angular velocity of the body frame is ω, the moment on the disk due to the Coriolis force can be computed as

$$\mathbf{M} = \mathbf{I}\omega \times \mathbf{p} \tag{3.2}$$

Since $\mathbf{I}$ is known we can compute the angular velocity ω by measuring angular velocity $\mathbf{p}$ and moment $\mathbf{M}$.

3.2 Data Fusion Methods

[25] gives a good introduction on visual inertial sensing. According to the authors, the methods for visual inertial fusion can be broadly categorized into two categories: loosely-coupled visual inertial fusion and tightly-coupled visual inertial fusion. Loosely-coupled visual inertial fusion systems treat the visual estimator and inertial estimator separately and run them at different rates. A filter is used to combine the data from visual and inertial estimators. The basic idea can be shown in figure 3.4. The images are first fed into a vision based localization system such as VSLAM or visual odometry (VO). Pose with scale ambiguity can be generated from the VSLAM/VO system. Later the pose with ambiguity is fused with the accelerometer and gyroscope

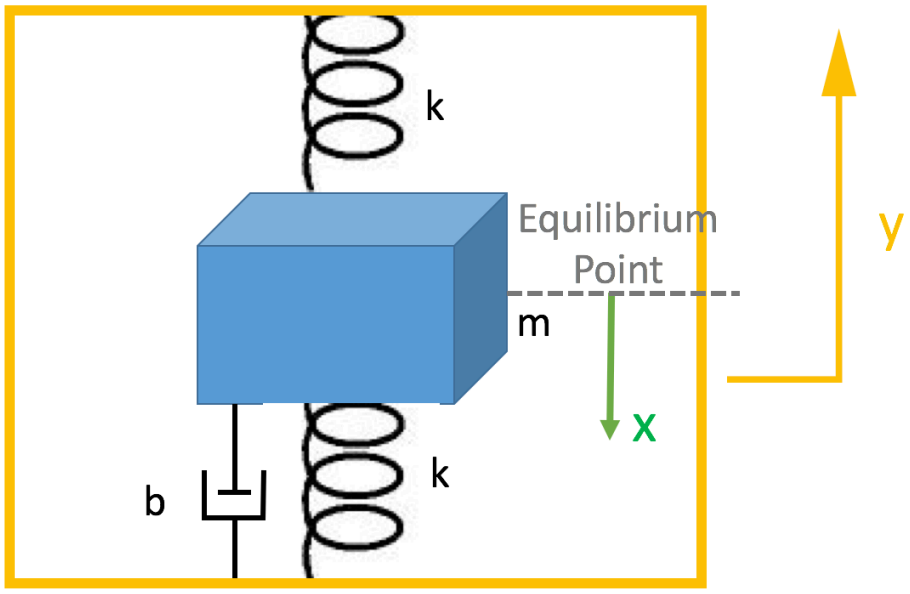

Figure 3.3: Physical principle of accelerometer

data from an IMU. The metric position, orientation, and biases in the IMU are estimated. Examples of loosely-coupled visual inertial fusion are [23] and [24].

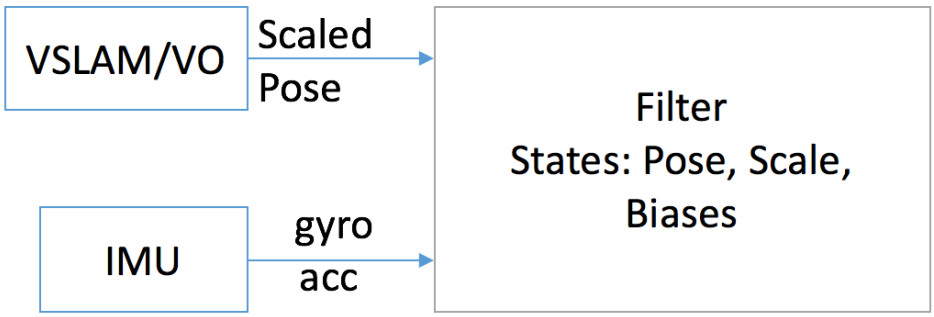

Figure 3.4: Loosely-coupled visual inertial fusion

Tightly-coupled visual inertial fusion systems combine the raw data both from vision (images) and IMU (gyroscope and accelerometer data) in a single filter. In the single statistical filter, the metric position, orientation, biases in the IMU, and position of features in the image are estimated. Examples of tightly-coupled visual inertial fusion are [26], [27].

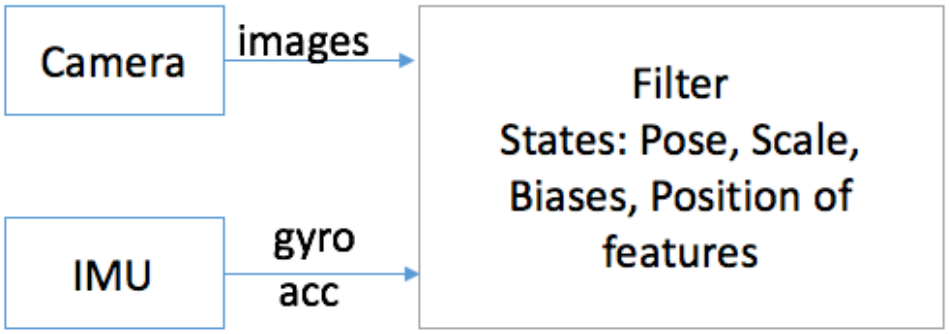

Figure 3.5: Tightly-coupled visual inertial fusion

There are advantages and disadvantages for both loosely-coupled visual inertial fusion and tightly-coupled visual inertial fusion. One of the biggest advantages of loosely-coupled visual inertial fusion is its modular property. The loosely-couple visual inertial systems treat the VSLAM/VO based pose estimation as a black box and therefore users can replace the VSLAM/VO method easily with the latest VSLAM/VO methods such as [28] and [29]. In addition, this structure provides convenience for system integration such as adding additional sensors. The problem with loosely-couple visual inertial fusion are: 1) while use visual and inertial sensors separately, some information of the correlations between sensors is missing 2) a failure detection mechanism is necessary to detect if the VSLAM/VO module is still working properly. Since tightly-coupled visual inertial fusion systems have maximal exploitation of sensing cues and take the correlations among internal states of different sensors into account, they can provide better accuracy and robustness [30]. The tightly-coupled visual inertial fusion systems can be much more complicated than loosely-coupled visual inertial fusion systems. In this thesis, we focus on loosely-coupled visual inertial fusion systems.

3.3 EKF based Loosely-coupled Visual Inertial Fusion

In this Section, an EKF based loosely-coupled sensor fusion system is presented. This sensor fusion framework is based on [5] and [24]. First, the linear Kalman filter and extend Kalman filter (EKF) are introduced. Then,

the application of EKF on loosely-coupled visual inertial sensor fusion is presented.

3.3.1 Kalman Filter

The Kalman filter is a popular algorithm to filter and fuse noisy sensor data and to produce state estimates. The estimated states in the Kalman filter tends to be more precise (has lower covariance) than the states computed based on a single measurement alone. Kalman filter is a recursive method, which means the estimation in current time step is based on the estimation in the previous time step. Also Kalman filter is one kind of Gaussian filter [31] which means the posteriors are presented as Gaussian random variables. Other assumptions of Kalman Filter are that the measurements are subject to Gaussian random noise and the system is linear. Under these assumptions, it can be shown that Kalman filter is an optimal estimator.

Assume the discrete time state transition model is:

$$\mathbf{x_k} = \mathbf{F_k x_{k-1}} + \mathbf{B_k u_k} + \mathbf{w_k} \tag{3.3}$$

where:

- $\mathbf{F_k}$: State transition matrix at the time step k.
- $\mathbf{x_k}$: States at time step k.
- $\mathbf{B_k}$: Control-input model at time step k.
- $\mathbf{u_k}$: Input at time step k.
- $\mathbf{w_k}$: Processing noise at time step k and with a zero mean normal distribution with covariance matrix $\mathbf{Q_k}$, i.e. $\mathbf{w_k} \sim \mathcal{N}(0, Q_k)$.

The measurement model is:

$$\mathbf{z_k} = \mathbf{H_k x_k} + \mathbf{v_k} \tag{3.4}$$

where:

- $\mathbf{z_k}$: Measurement at time step k.

- $\mathbf{H_k}$: Output model at time step k.
- $\mathbf{v_k}$: Measurement noise at time step k with a zero mean normal distribution with covariance matrix $\mathbf{R_k}$ (i.e. $\mathbf{w_k} \sim \mathcal{N}(0, R_k)$).

Also denote:

- $\hat{\mathbf{x}}_{k|k-1}$: Prediction at time step k.
- $\mathbf{P}_{k|k-1}$: Prediction covariance at time step k.
- $\hat{\mathbf{x}}_{k|k}$: Posterior at time step k, which means the estimation after incorporating the measurement at time step k.
- $\mathbf{P}_{k|k}$: Posterior covariance at time step k.

The Kalman filter has a two-step structure:

1. Prediction:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{F_k}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B_k}\mathbf{u_k} \\ \mathbf{P}_{k|k-1} &= \mathbf{F_k}\hat{\mathbf{P}}_{k-1|k-1}\mathbf{F_k}^{\mathbf{T}} + \mathbf{Q_k} \end{aligned} \tag{3.5}$$

2. Correction:

$$\begin{aligned} \mathbf{S_k} &= \mathbf{H_k}\mathbf{P}_{k|k-1}\mathbf{H_k}^{\mathbf{T}} + R_k \\ \mathbf{K_k} &= \mathbf{P}_{k|k-1}\mathbf{H_k}^{\mathbf{T}}\mathbf{S_k}^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K_k}(\mathbf{z_k} - \mathbf{H_k}\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_{k|k} &= (I - \mathbf{K_k}\mathbf{H_k})\mathbf{P}_{k|k-1} \end{aligned} \tag{3.6}$$

Note in the prediction step, the covariance of states increases and in the correction step, the covariance of states decreases.

3.3.2 Extended Kalman Filter

In order to apply the Kalman filter in nonlinear systems

$$\begin{aligned} \mathbf{x_k} &= \mathbf{f}(\mathbf{x_{k-1}}, \mathbf{u_k}) + \mathbf{w_k} \\ \mathbf{z_k} &= \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k} \end{aligned} \tag{3.7}$$

we need to linearize nonlinear function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{h}(\mathbf{x})$ in equation 3.7 with respect current states and inputs. This results in:

$$\begin{aligned} \mathbf{F_k} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{\mathbf{k}|\mathbf{k-1}}, \mathbf{u_k}} \\ \mathbf{H_k} &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{\mathbf{k}|\mathbf{k-1}}}. \end{aligned} \tag{3.8}$$

Then equations 3.5 and 3.6 will become:

1. Prediction:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u_k}) \\ \mathbf{P}_{k|k-1} &= \mathbf{F_k}\hat{\mathbf{P}}_{k-1|k-1}{\mathbf{F_k}}^{\mathbf{T}} + \mathbf{Q_k} \end{aligned} \tag{3.9}$$

2. Correction:

$$\begin{aligned} \mathbf{S_k} &= \mathbf{H_k}\mathbf{P}_{k|k-1}{\mathbf{H_k}}^{\mathbf{T}} + R_k \\ \mathbf{K_k} &= \mathbf{P}_{k|k-1}{\mathbf{H_k}}^{\mathbf{T}}{\mathbf{S_k}}^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K_k}(\mathbf{z_k} - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})) \\ \mathbf{P}_{k|k} &= (I - \mathbf{K_k}\mathbf{H_k})\mathbf{P}_{k|k-1} \end{aligned} \tag{3.10}$$

3.3.3 Visual Inertial Loosely-coupled Fusion with EKF

In this section, we briefly present the idea of fusing VSLAM output with IMU data using the EKF based on [5] and [3]. To begin with, we specify the coordinate system used in this section. There are four frames:

- Vision frame: the reference frame in PTAM. The vision frame is denoted as $\mathcal{V}$
- World frame: gravity aligned frame. The origin's position is the same as the vision frame. The world frame is denoted as $\mathcal{W}$.
- Body frame: the frame attached to the IMU. This frame will rotate when the UAV rotates. Body frame is denoted as $\mathcal{B}$.
- Camera frame: the frame attached to camera. The camera frame is denoted as $\mathcal{C}$.

The states we estimate are: the position and velocity of the IMU in the world frame ($p_{\mathcal{B}}^{\mathcal{W}}$, $v_{\mathcal{B}}^{\mathcal{W}}$), the rotation quaternion from the world frame to IMU frame $q_{\mathcal{W}}^{\mathcal{B}}$, the acceleration bias b_a, the gyroscope bias b_ω, the rotation quaternion from the vision frame to the world frame $q_{\mathcal{V}}^{\mathcal{W}}$, the scale λ, the rotation quaternion from body frame to camera frame $q_{\mathcal{B}}^{\mathcal{C}}$ and the position of the camera frame in the body frame $p_{\mathcal{C}}^{\mathcal{B}}$. Measurements from the IMU are a_m and ω_m with noise n_a and n_ω The system dynamics can be presented as:

$$\begin{aligned}
\dot{p}_{\mathcal{B}}^{\mathcal{W}} &= v_{\mathcal{B}}^{\mathcal{W}} \\
\dot{v}_{\mathcal{B}}^{\mathcal{W}} &= R_{\mathcal{B}}^{\mathcal{W}}(a_m - b_a - n_a) - g \\
\dot{q}_{\mathcal{W}}^{\mathcal{B}} &= \frac{1}{2}(\omega_m - b_\omega - n_\omega)q_{\mathcal{W}}^{\mathcal{B}} \\
\dot{b}_\omega &= n_{b_\omega}, \dot{b}_a = n_{b_a}, \dot{\lambda} = 0, \dot{q}_{\mathcal{V}}^{\mathcal{W}} = 0.
\end{aligned} \tag{3.11}$$

where R_a^b is the corresponding rotation matrix of quaternion q_a^b.

Denote the measurement of unscaled position as z_p and the measured orientation as z_q. The measurement is:

$$\begin{aligned}
z_p = p_{\mathcal{C}}^{\mathcal{V}} &= R_{\mathcal{W}}^{\mathcal{V}}(p_{\mathcal{W}}^{\mathcal{B}} + R_{\mathcal{B}}^{\mathcal{W}} p_{\mathcal{C}}^{\mathcal{B}})\lambda \\
z_q = q_{\mathcal{V}}^{\mathcal{C}} &= q_{\mathcal{B}}^{\mathcal{C}} \bigotimes q_{\mathcal{B}}^{\mathcal{W}} \bigotimes q_{\mathcal{W}}^{\mathcal{V}},
\end{aligned} \tag{3.12}$$

where $\bigotimes$ represents the quaternion multiplication.

The system can be linearized and discretized to fit into the form shown in 3.3.2. Then, EKF can be used to solve the estimation problem.

3.4 Implementation and Experimental Results

In this section, the implementation of the framework shown in [5] and [3] is presented. In Section 3.4.1, the hardware system is introduced. In Section 3.4.2, a real experiment is performed and the estimation result is shown to prove the success of the implementation.

3.4.1 Hardware Implementation

We implemented the sensor fusion framework[1] on an AscTec Firefly UAV (figure 3.6). The hexacopter is redundant and offers vibration damped slots for various payloads. It is equipped with an AscTec MasterMind with a high performance 1.7 GHz third Generation Intel Core i7 processor and multiple USB 2.0 ports to communicate with the payload.

To apply the PTAM method discussed in the previous section, we fixed a downward-facing MatrixVision Bluefox camera on the Firefly. The camera sends images to the Mastermind at about 40 frames per second.

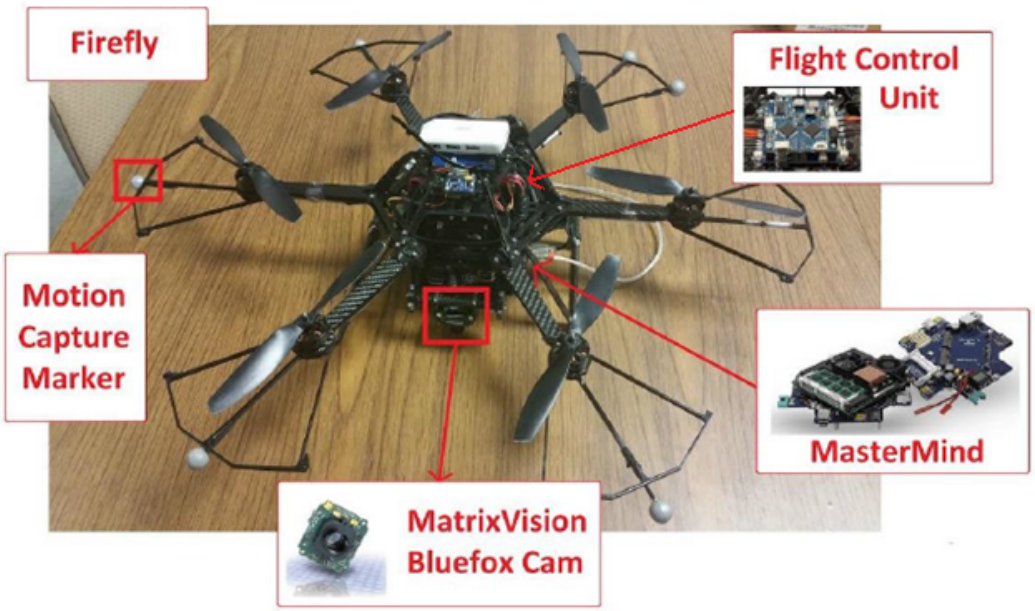

Figure 3.6: Experimental setup of the AscTec Firefly equipped with the Bluefox camera, MasterMind board, flight control unit, and motion capture markers.

The basic architecture of the sensor fusion framework is shown in figure 3.7. The state prediction is performed on the flight control unit (microcontroller), which contains the IMU. The prediction is running at 1000 Hz. The predicted states are sent to the onboard computer to perform the covariance prediction and update. When the states are updated by incorporating the data from VSLAM, they are sent back to the flight control unit for UAV control.

[1] http://wiki.ros.org/ethzasl_sensor_fusion

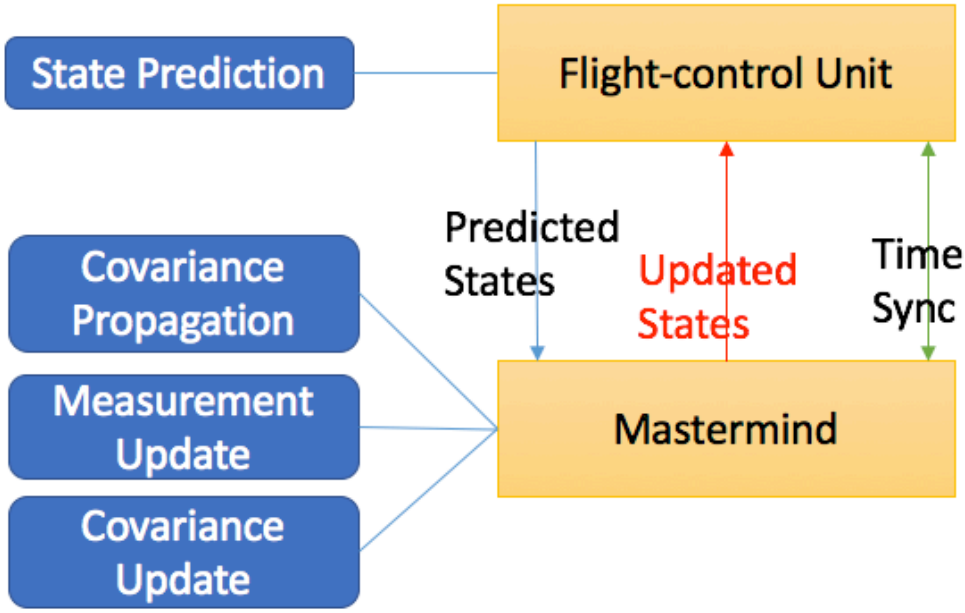

Figure 3.7: Architecture of sensor fusion framework

3.4.2 Experimental Results

To verify the position and attitude estimates from PTAM and the sensor fusion, we use a motion capture system to provide a ground truth. The motion capture system consists of 24 OptiTrack Flex 3 cameras that capture pose data at 100 Hz with an centimeter level accuracy. We perform 5 separate tests and the overall testing time is 2260.2 seconds.

First we test the position estimation performance when the initialization error of scale is small (less than 0.1). The real scale is estimated by performing least square estimation. Denote the coordinate from the motion capture at time step i as $\mathbf{x}_i^{mocap} \in \mathbb{R}^3$ and the coordinate from the PTAM at the same time as $\mathbf{x}_i^{ptam} \in \mathbb{R}^3$, we solve the scale λ by solving the minimization problem:

$$\lambda^* = \underset{\lambda}{argmin} \sum_i \left\| \mathbf{x}_i^{ptam} - \lambda \mathbf{x}_i^{mocap} \right\| = \frac{\sum_i (\mathbf{x}_i^{ptam})^{\mathbf{T}} \mathbf{x}_i^{mocap}}{\sum_i (\mathbf{x}_i^{mocap})^{\mathbf{T}} \mathbf{x}_i^{mocap}}. \tag{3.13}$$

We use test 1 as an example. The initialization error of scale is 0 and the estimation result is shown as figure 3.8. The RMS error of scale estimation is

Test	Duration (s)	PTAM estimation			Sensor fusion estimation			
		x	y	z	scale	x (m)	y (m)	z (m)
1	185	0.0789	0.0985	0.0511	0.0440	0.0446	0.0622	0.0154
2	211.2	0.1056	0.1064	0.0440	0.0221	0.0744	0.0597	0.0263
3	692.4	0.4381	0.3557	0.5403	0.0134	0.0574	0.0668	0.0520
4	604.0	0.1527	0.1399	0.2842	0.0655	0.0584	0.0511	0.0533
5	567.6	0.1368	0.1047	0.0769	0.0682	0.0553	0.0540	0.0406

Table 3.1: RMS error in position estimation, scale initialization error is 0

0.0440. The estimation results of x,y,z position are shown as figure 3.9, 3.10, and 3.11. We can see, the position estimation is more accurate after fusing with IMU data. Moreover, notice that from 125th second to 140th second, although PTAM experiences a short time failure in position estimation, the position estimation from visual inertial fusion system can still track the actual motion of the UAV. The RMS error in position estimation of both PTAM and visual inertial fusion is shown as table 3.1. We can see, when the scale is initialized accurately, by fusing the output of PTAM with IMU data we can achieve better precision than only using the PTAM output.

Then we perform visual inertial fusion to the same data but with greater initialization error (0.4 instead of 0) in scale. Again, we use test 1 as an example. Figure 3.12 shows the convergence of scale. The corresponding position estimation is shown as figure 3.13, 3.14, and 3.15. The RMS error in position estimation is shown as table 3.2. Notice in the table the PTAM estimation error is not shown since it is the same as in table 3.1. Comparing table 3.1 and 3.2 we can see, while the scale is converging, scale estimation error can cause position estimation error. In the next chapter, scale estimation will be discussed.

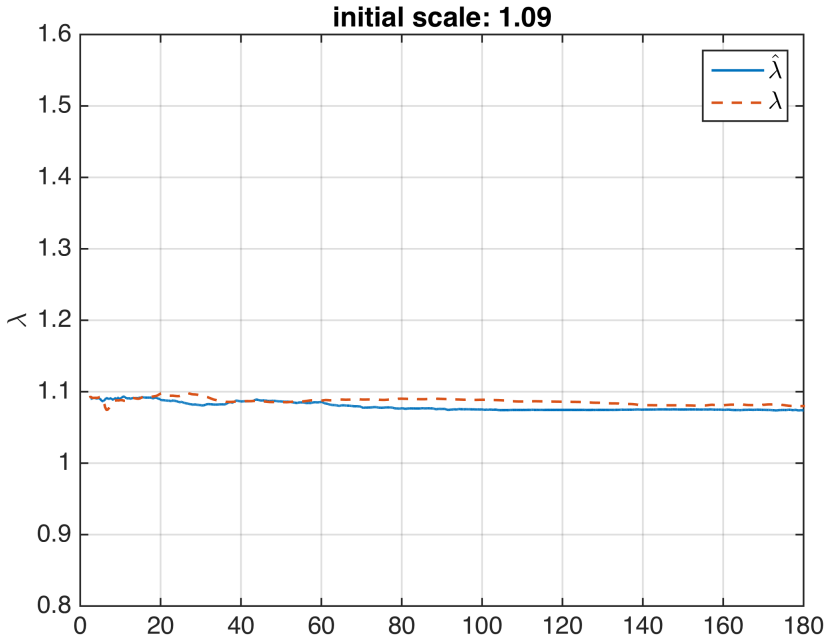

Figure 3.8: Estimation of scale when the scale initialization error is 0

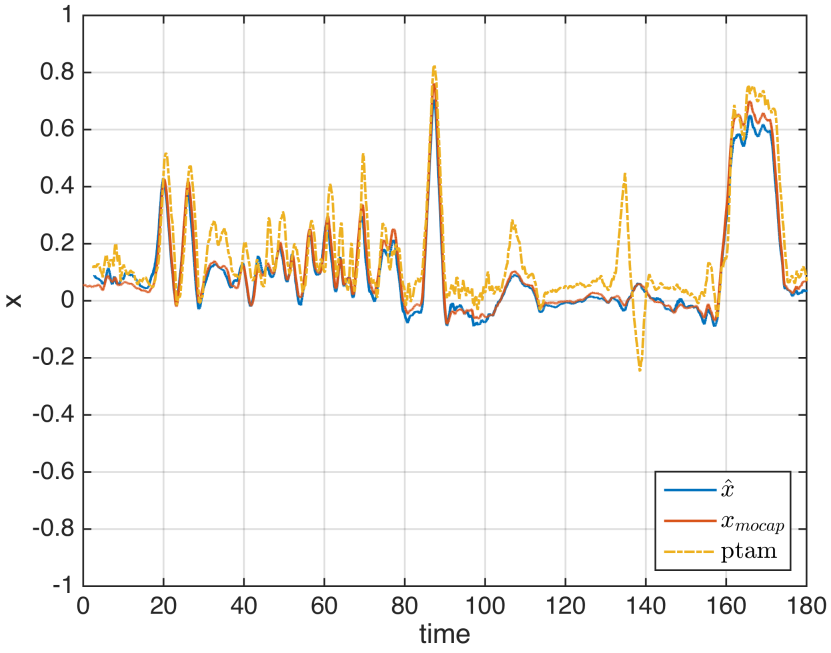

Figure 3.9: Estimation of position in x direction when the scale initialization error is 0

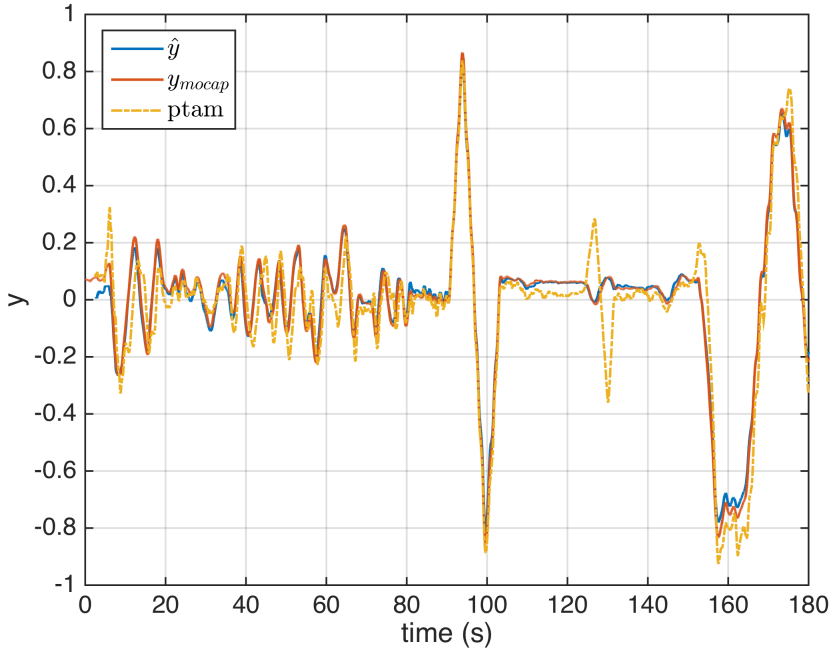

Figure 3.10: Estimation of position in y direction when the scale initialization error is 0

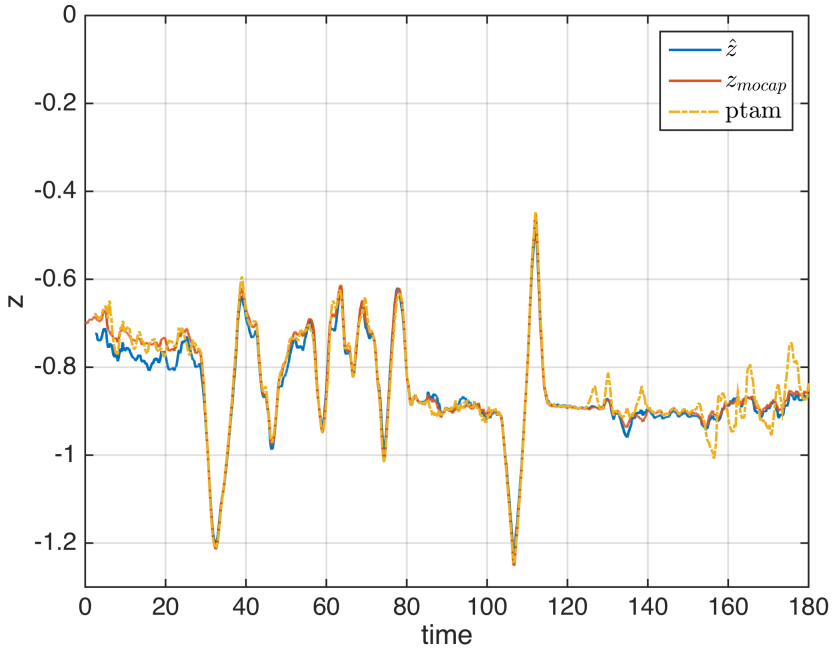

Figure 3.11: Estimation of position in z direction when the scale initialization error is 0

Test	Duration (s)	Sensor fusion estimation			
		scale	x (m)	y (m)	z (m)
1	185	0.3082	0.0481	0.0974	0.0514
2	211.2	0.3164	0.0844	0.1253	0.0522
3	692.4	0.1117	0.0639	0.1114	0.1235
4	604.0	0.2217	0.1533	0.1153	0.1620
5	567.6	0.2049	0.0610	0.0909	0.1540

Table 3.2: RMS error in position estimation, scale initialization error is 0.4

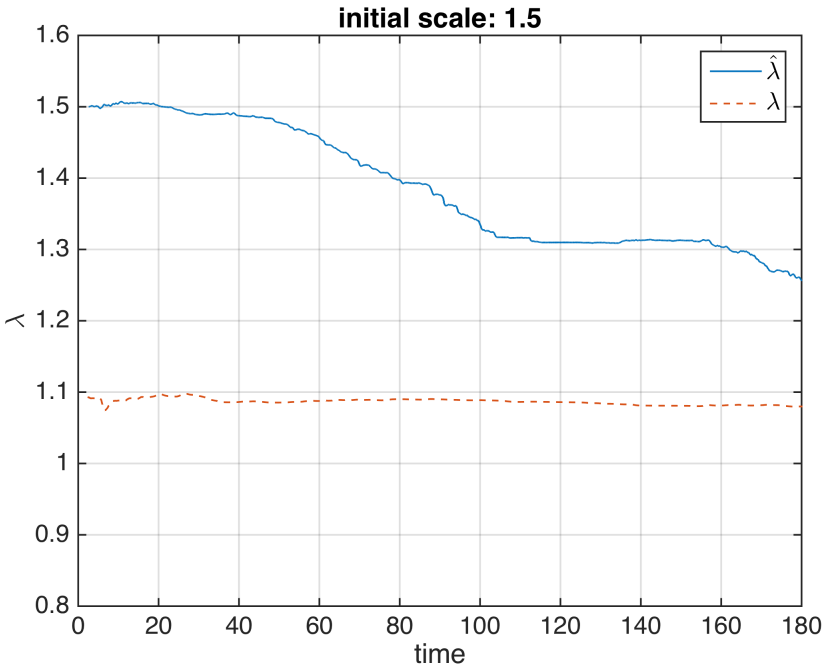

Figure 3.12: Estimation of scale when the scale initialization error is 0.4

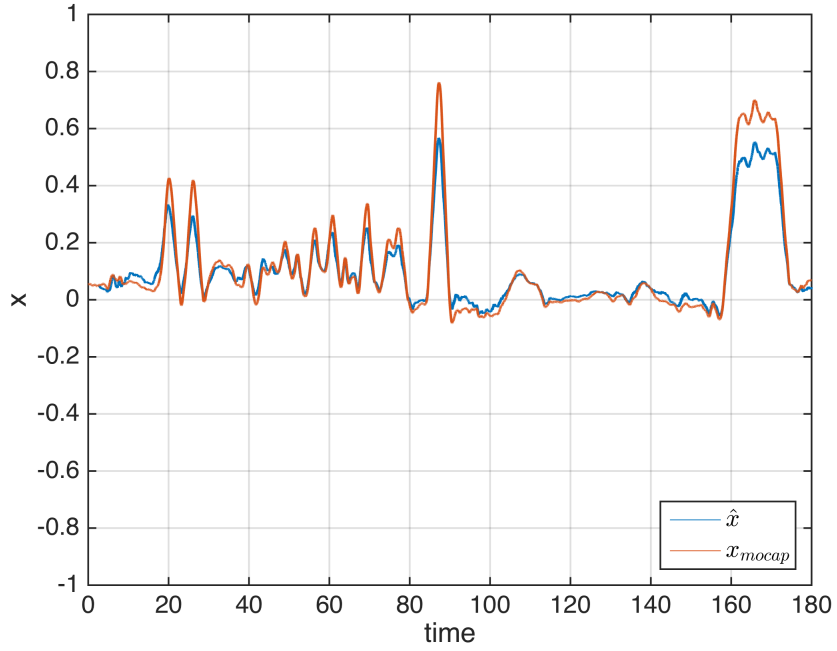

Figure 3.13: Estimation of position in z direction when the scale initialization error is 0.4

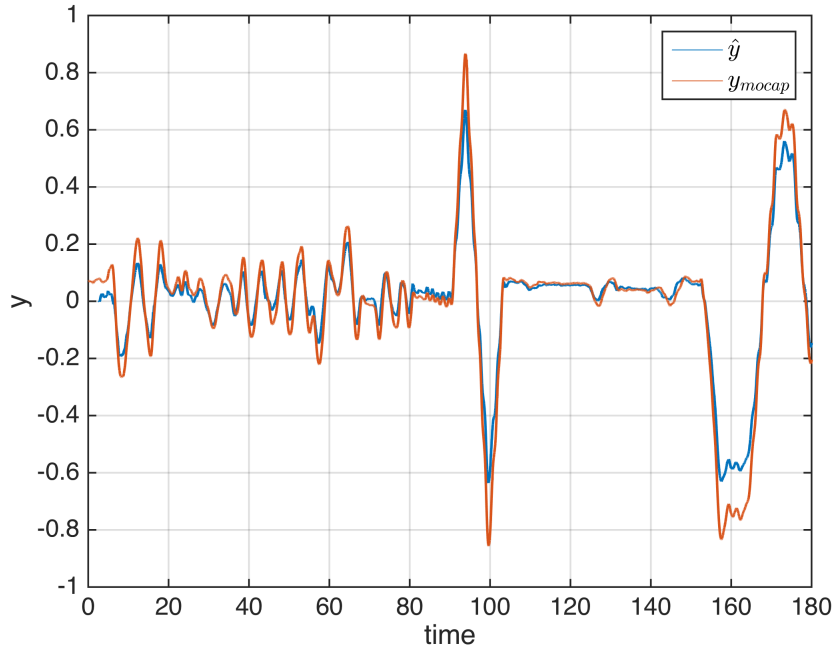

Figure 3.14: Estimation of position in z direction when the scale initialization error is 0.4

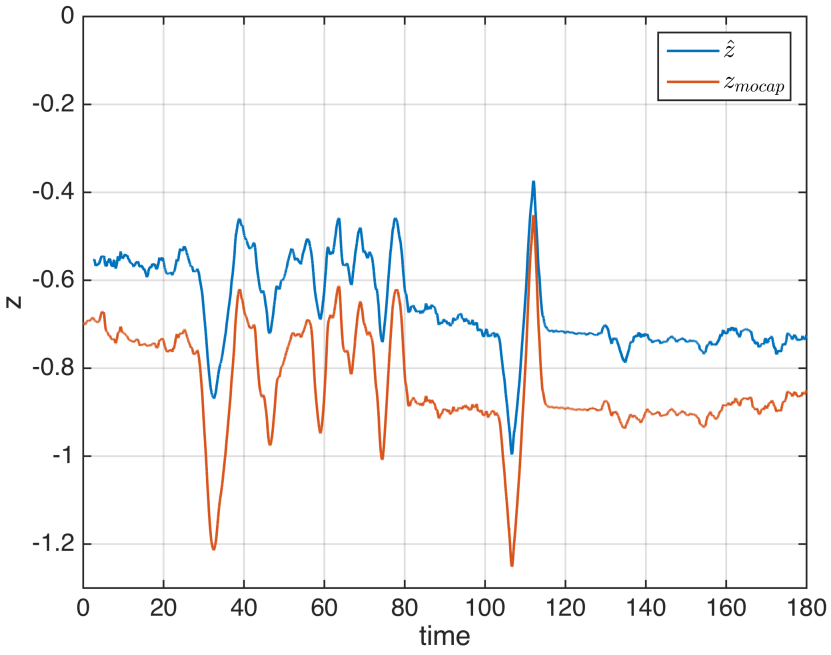

Figure 3.15: Estimation of position in z direction when the scale initialization error is 0.4

CHAPTER 4

SCALE ESTIMATION

As discussed in Chapter 2 and 3, fusing IMU data with the output from VSLAM systems can resolve the ambiguity of scale. In addition, experimental results have shown that the estimation error of scale may result in an error in the estimation of position. When applying EKF to perform state estimation, the nonlinear system is linearized with respect to the estimated state, which is called as operating point. In this chapter, the relationship between the scale estimation performance and the operating point of scale will be presented. To the best of the author's knowledge, this relationship has not been presented and analyzed before. The author believes understanding this relationship will be beneficial in designing scale estimators which have more consistent performance. We will first present the relationship in the EKF based scale estimation. A simplified model, which can capture the relation, is then established to simplify the analysis. Lastly, the operation point's effects on the innovation term of the EKF will be discussed with simulation results.

4.1 Problem Formulation

The scale λ can be defined as:

$$\lambda = \frac{Motion_{slam}}{Motion_{real}} \tag{4.1}$$

$Motion_{slam}$ represents the displacement of the camera estimated in the VSLAM system, and $Motion_{real}$ represents the actual displacement which is in SI units. In VSLAM systems, the distance between the first and the second keyframes is usually assumed as unit distance in order to solve for the essential matrix. The actual displacement, however, can be of any distance. This

means theoretically the magnitude scale can be any number between 0 and infinity.

Before fusing the VSLAM output with the IMU data to estimate the scale, an initial guess of the scale has to be given. Figure 4.1, 4.2, and 4.3 show the convergence behavior of scale with different initial guesses. As can be seen, the convergence rate becomes slower when the magnitude of scale guess is greater.

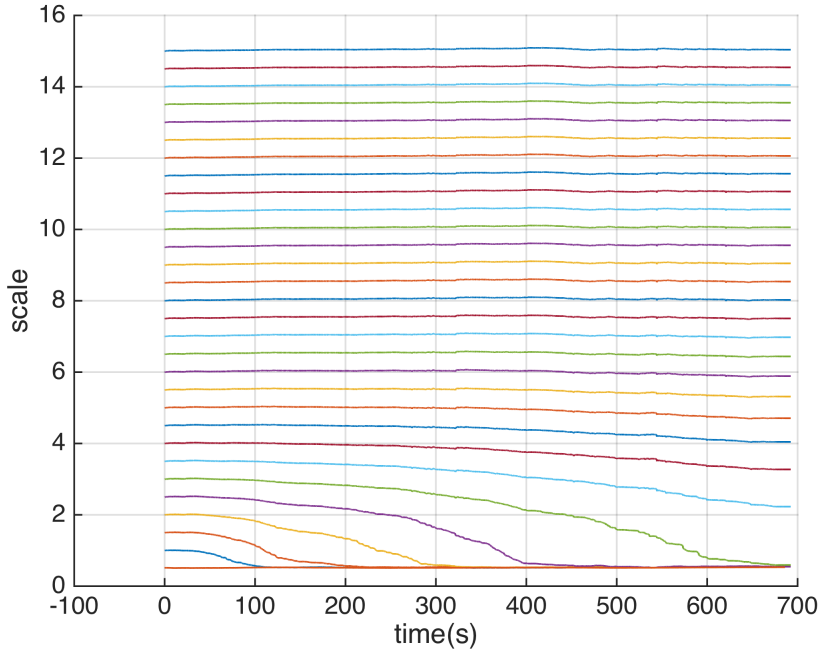

Figure 4.1: Scale estimation with different initial guesses (experiment 1)

In order to clarify the relationship between convergence rate and the operating point of the system. With the data in experiment 1, we fix the error in the initial guesses (i.e. keep the initial guess 0.5 lower than the real scale) and vary the real scale from 0.52 to 2.6 with step size 0.52. The convergence can be seen in figure 4.4. The dot dash lines represent the real values of scale. The scale is considered to be converged when the estimation error decreases below 0.05. We plot the relationship between the real scale and the convergence time in figure 4.5. As the real data shows, while applying the EKF to loosely-coupled visual inertial fusion, the convergence rate of scale decreases when the operating point of scale increases. That is to say, with the same external excitation, it will take longer time for scale to converge from $a + \underline{\Delta\lambda}$

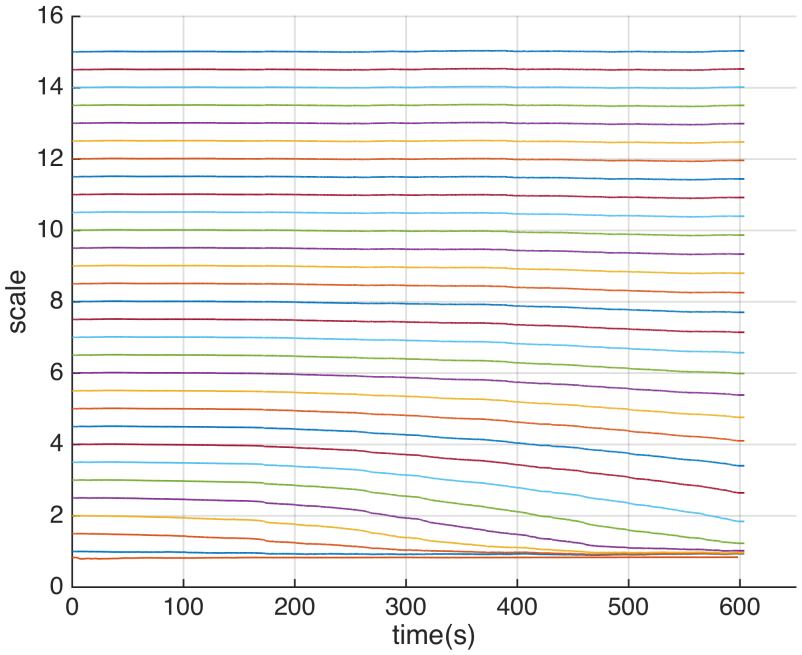

Figure 4.2: Scale estimation with different initial guesses (experiment 2)

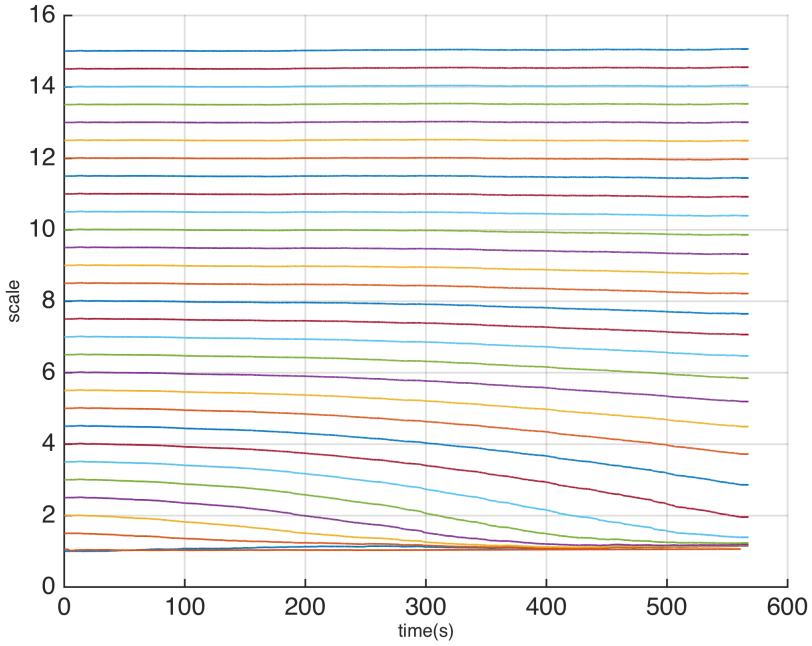

Figure 4.3: Scale estimation with different initial guesses (experiment 3)

to a than $b + \underline{\Delta\lambda}$ to b, where a and b are positive numbers subject to $a > b$ and $\underline{\Delta\lambda}$ is the initial error of scale. In the rest of this chapter, we analyze and explain this relationship.

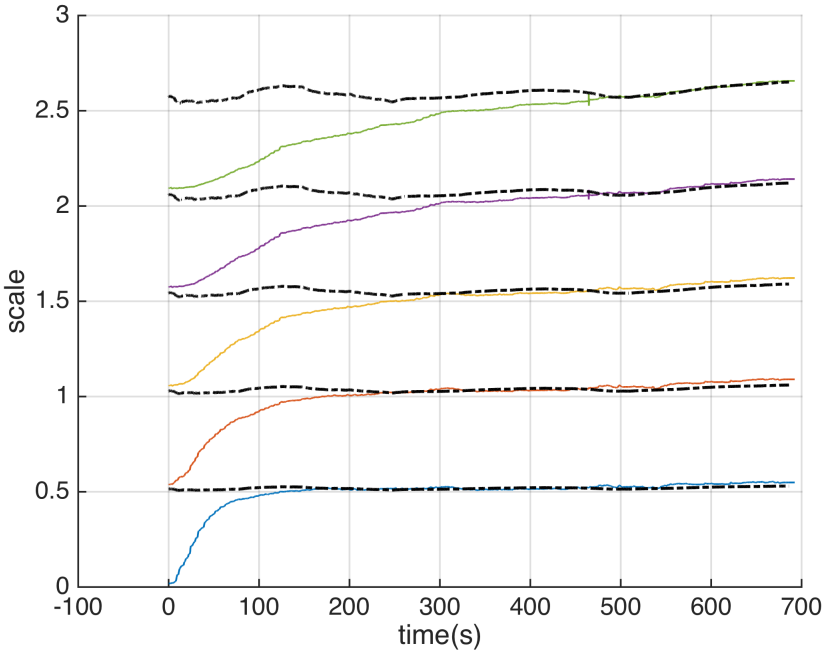

Figure 4.4: Scale estimation with different scale magnitude

4.2 Model of Scale Estimation Problem

We propose a model for the scale estimation problem in this section. Simulation results are shown to prove this simplified model captures the relationship between the operating point of the system and convergence rate. We review the continuous time extended Kalman filter first, because the continuous time EKF will be used in the analysis in the remainder of this chapter. In addition, the relationship between the Kalman gain in the continuous time EKF and the Kalman gain in discrete time EKF is presented.

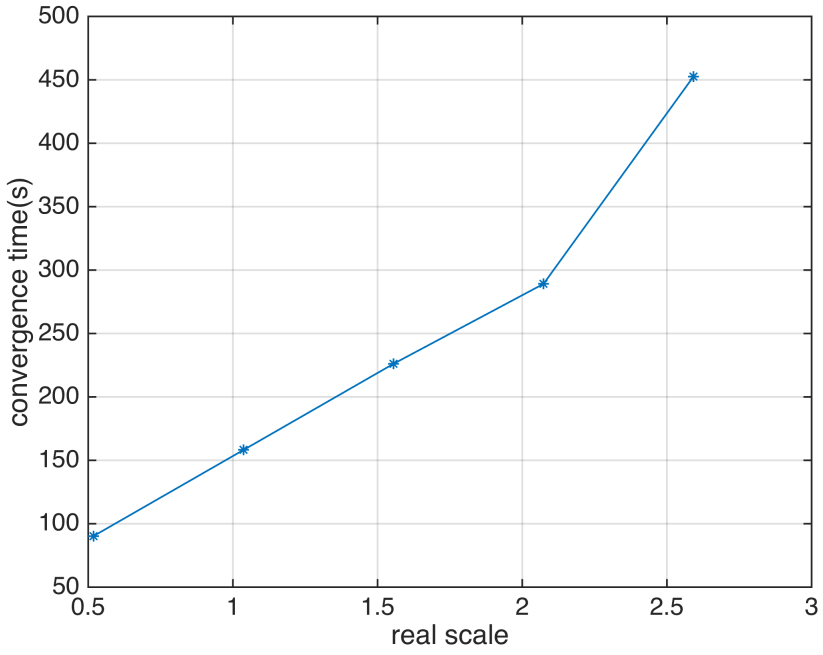

Figure 4.5: Relationship between the magnitude of scale and convergence time

4.2.1 Continuous-time Extended Kalman Filter

With a dynamic system:

$$\begin{aligned} \dot{x} &= f(x,u) + w, \quad w \sim N(0,Q) \\ y &= h(x) + v, \quad v \sim N(0,R) \end{aligned} \tag{4.2}$$

The prediction and update are coupled in the continuous time EKF, which is different from the discrete time EKF. The way to solve the continuous time EKF is presented as follows

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x},u) + K(z - h(\hat{x})) \\ \dot{P} &= FP + PF^T - KHP + Q \\ K &= PH^T R^{-1} \\ F &= \frac{\partial f}{\partial x}|_{\hat{x},u}; \quad H = \frac{\partial h}{\partial x}|_{\hat{x}} \end{aligned} \tag{4.3}$$

where $\hat{x}$ is the estimation of state, z is the actual measurement, P is the covariance matrix of the states and K is the Kalman gain.

From [32], denoting the Kalman gain computed in the discrete time system as K_d and the step size in time discretization is Δt, we have

$$K_d \approx K\Delta t \tag{4.4}$$

assuming Δt is sufficiently small. Note that the sign of K_d and K is the same.

4.2.2 Model of scale estimation

The proposed model of the scale estimation problem can be written as

$$\begin{aligned} \dot{x} &= v \\ \dot{\lambda} &= 0 \\ z &= \lambda x. \end{aligned} \tag{4.5}$$

The physical meaning of x is the position of a visual inertial system which is subject to one-dimensional motion. Assume the acceleration can be precisely measured, and the velocity is precisely initialized. This means the velocity of the visual inertial system v can be measured precisely. We use λ to denote the scale and z to denote the measurement. The real scale can be represented as c, which is a constant. We initialize $\hat{\lambda}$ at $c + \underline{\Delta\lambda}$ where the initial error $\underline{\Delta\lambda}$ in scale estimation is sufficiently small (e.g. 0.5).

Denote the position estimation error as $\Delta x = x - \hat{x}$ and scale estimation error as $\Delta\lambda = \lambda - \hat{\lambda}$. To analyze the effect of operating point of scale on the performance of scale estimation, we assume we have accurate guess of the initial position

$$\hat{x}(0) = x(0), \tag{4.6}$$

we also assume we know the processing noise covariance matrix Q and measurement noise covariance matrix R.

4.2.3 Simulation result

In the simulation, without loss of generality, the trajectory of the visual-inertial system can be specified as:

$$x = \sin(0.2t) \tag{4.7}$$

The simulation time spans from 0 to 100 s. The initialization error $\underline{\Delta\lambda}$ is set as $+0.5$. Assuming a precisely initialized position, real scale c varies from 1 to 15 with step size 2. The convergence of scale estimation error ($\Delta\lambda$) is shown in figure 4.6. Note that the operating point of scale $\hat{\lambda}$ is positively related to real scale c (i.e. greater real scale c indicates greater $\hat{\lambda}$). We can use the c to represent the level of the magnitude of scale (we also call it scale level in this chapter). With the convergence criteria as 0.1, the relationship between real scale "c" and convergence time can be shown as figure 4.7.

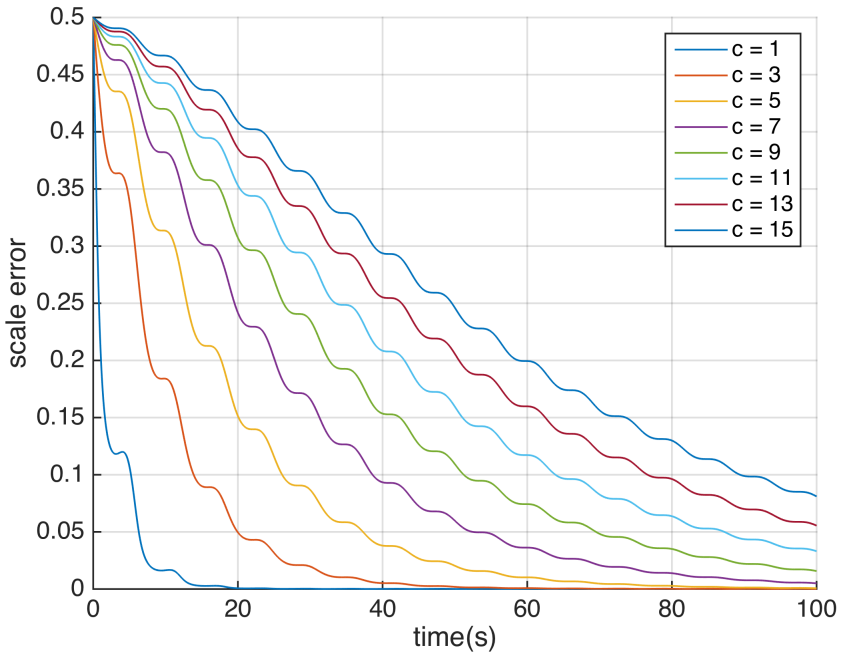

Figure 4.6: Convergence behavior of scale with different real scale values. Lines with different colors correspond to different real scale values

We can see the scale convergence rate decreases as the magnitude of operating point ($\hat{\lambda}$) increases, this coincide with the relationship shown in the real data.

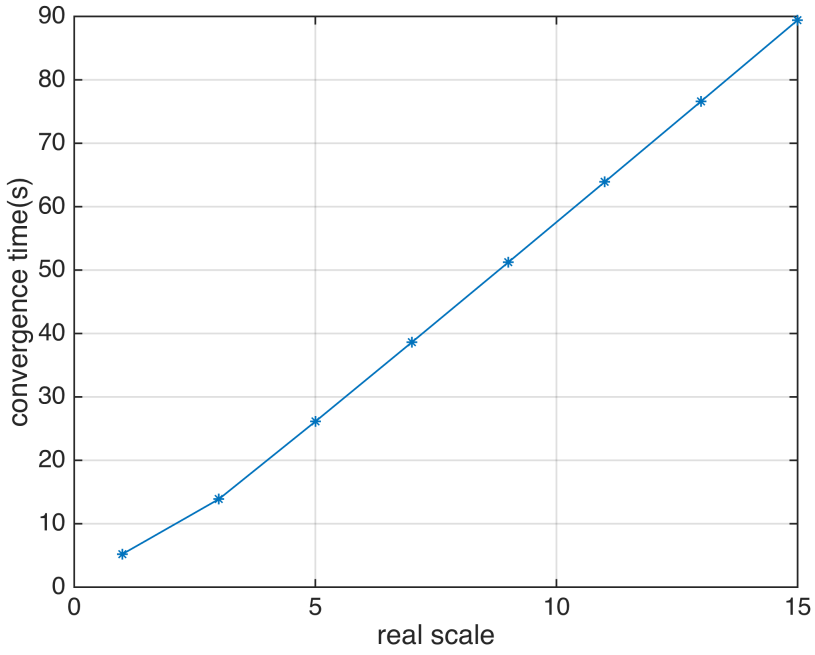

Figure 4.7: Convergence time of scale with different real scale values

If we combine the proposed equation 4.5 and equation 4.3, we have

$$\begin{aligned} \dot{\hat{x}} &= v + k_1(z - \hat{\lambda}\hat{x}) \\ \dot{\hat{\lambda}} &= k_2(z - \hat{\lambda}\hat{x}). \end{aligned} \tag{4.8}$$

Recall the definition of estimation error Δx and $\Delta\lambda$ is

$$\begin{aligned} \Delta x &= x - \hat{x} \\ \Delta\lambda &= \lambda - \hat{\lambda}, \end{aligned} \tag{4.9}$$

we have

$$\begin{aligned} \Delta\dot{x} &= -k_1(z - \hat{\lambda}\hat{x}) \\ \Delta\dot{\lambda} &= -k_2(z - \hat{\lambda}\hat{x}). \end{aligned} \tag{4.10}$$

We can see the convergence of $\Delta\lambda$ has two contributors:

- The Kalman gain.
- The innovation term $z - \hat{\lambda}\hat{x}$.

In the next section, we will perform analysis and simulation to explain the

effects of the operating point of scale ($\hat{\lambda}$) on the innovation term.

4.3 Operating Point of Scale's Effects on Innovation Term

In order to analyze the effects of the innovation term, we first fix the Kalman gain. We precompute the time variant Kalman gain when the real scale "c" is equal to 15 (figure 4.8 and 4.9) and apply it to different situations with different "c". The relationship between the convergence time and scale level is shown as figure 4.10 and 4.11.

As we can see, when the scale level increases the convergence rate for scale still decreases. Figure 4.12 shows the effect of scale level on the innovation term, different colors represent different scale levels. We can see that the smaller the scale level (also the magnitude of the operating point of scale), the greater the amplitude of the innovation term. Greater innovation term will result in larger output injection in the estimator and a faster convergence rate.

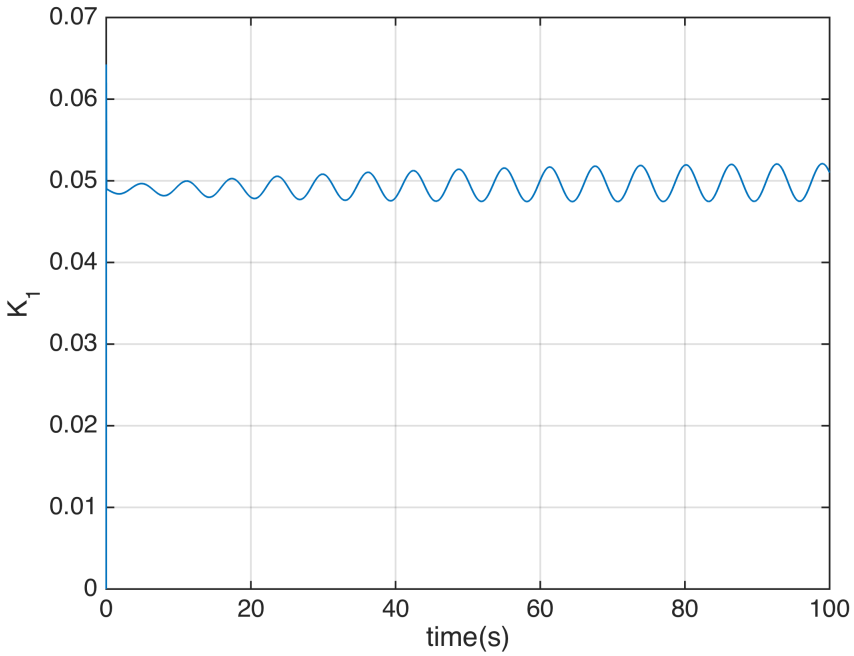

Figure 4.8: Precomputed Kalman gain k_1

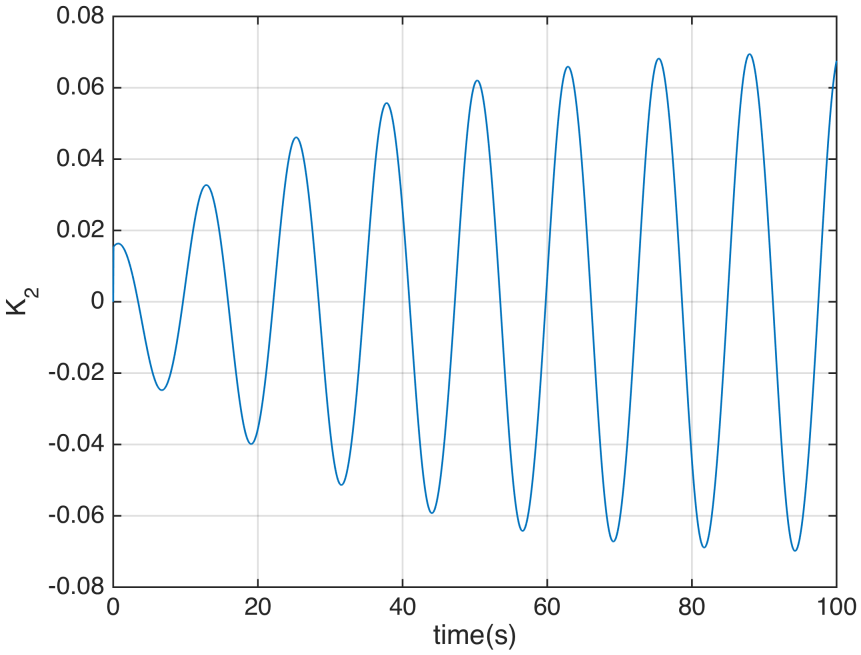

Figure 4.9: Precomputed Kalman gain k_2

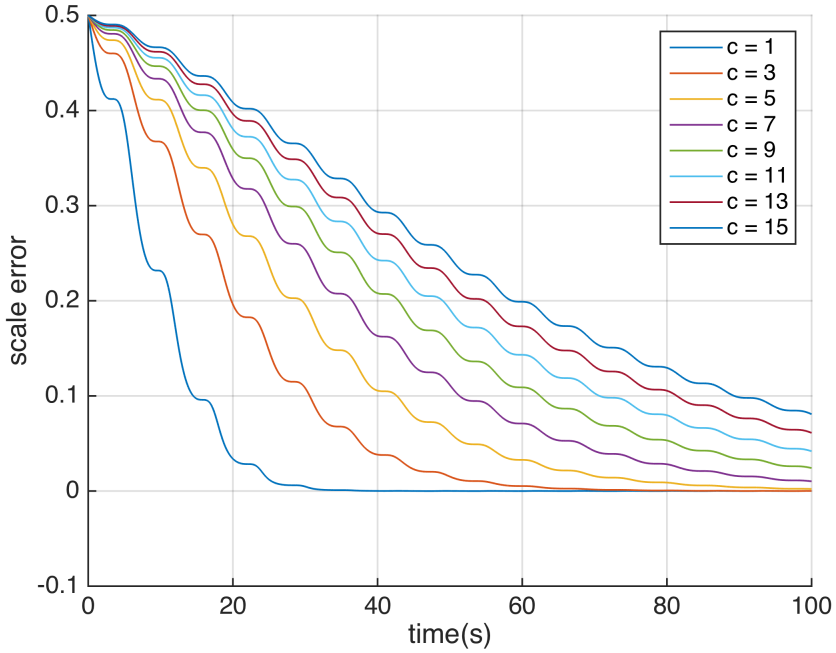

Figure 4.10: Convergence behavior of scale with different real scale values, but the same Kalman gain. Lines with different colors correspond to different real scale values

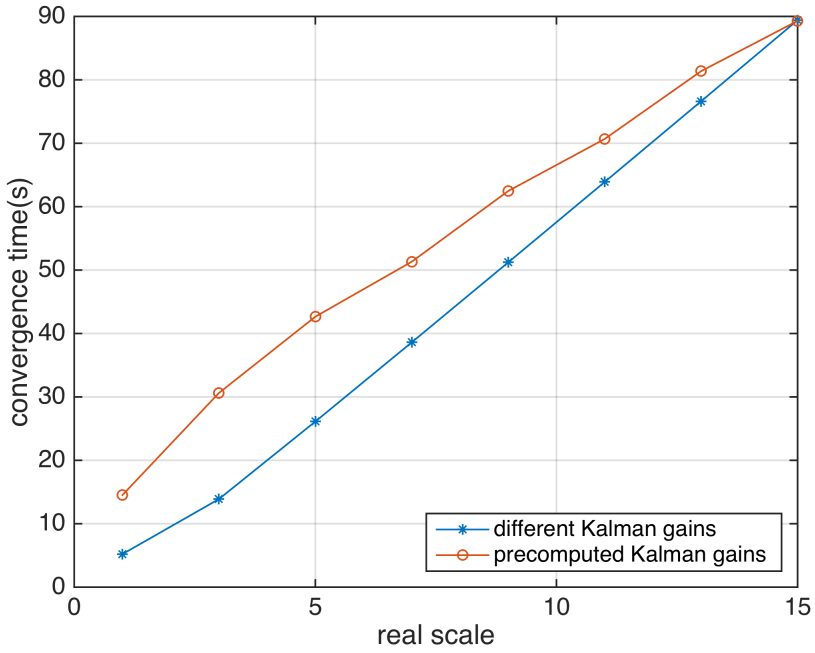

Figure 4.11: Convergence time of the systems using different Kalman gains and the same Kalman gain

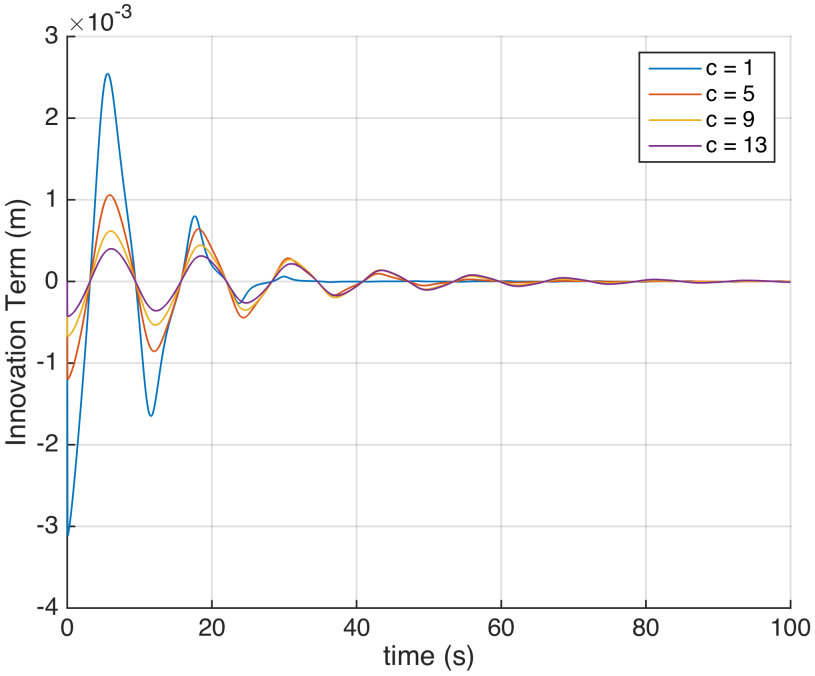

Figure 4.12: Effect of scale level on innovation term. Lines with different colors correspond to different real scale values

To sum up, in this chapter we present a relationship between the convergence rate of scale estimation and the operating point of the system. In particular, we find that greater magnitude of scale will cause lower convergence rate in the scale estimation. From the simulation, we found one of the reasons of this relationship is greater magnitude of scale will result in smaller amplitude of the innovation term.

CHAPTER 5

CONCLUSIONS

In this thesis, we have presented a system which uses a monocular camera and an IMU to navigate an aerial robot in an previously unknown area. Parallel Tracking and Mapping (PTAM) is used provide the scaled position and orientation of the aerial robot. The output of PTAM is then fused with IMU data to resolve the scale ambiguity result from the nature of monocular vision. We perform flight tests to show the improvement in estimation by fusing IMU data. We have also shown that, when using EKF to perform loosely-coupled visual inertial fusion, the convergence rate of scale depends on the operating point of scale. When the magnitude of the operating point of scale increases, the convergence rate of scale decreases. From the simulation, we have shown this relationship is because when the operating point of scale increases, the amplitude of innovation term in the EKF becomes smaller. In the future we shall explore utilizing this property to improve the scale estimation in visual inertial sensor fusion.

REFERENCES

[1] D. Shaw, "Disaster drones: How robot teams can help in a crisis," 2012. [Online]. Available: http://www.bbc.com/news/technology-18581883

[2] W. Knight, "New boss on construction sites is a drone," 2015. [Online]. Available: http://www.technologyreview.com/news/540836/new-boss-on-construction-sites-is-a-drone/

[3] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[4] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, p. 0278364911434236, 2012.

[5] S. M. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, ETH Zurich, 2012.

[6] M. W. Achtelik, "Advanced closed loop visual navigation for micro aerial vehicles," Ph.D. dissertation, ETH Zurich, 2014.

[7] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3400–3407.

[8] K. Ling, D. Chow, A. Das, and S. L. Waslander, "Autonomous maritime landings for low-cost vtol aerial vehicles," in *Computer and Robot Vision (CRV), 2014 Canadian Conference on*. IEEE, 2014, pp. 32–39.

[9] 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*. IEEE, 2007, pp. 225–234.

[10] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[12] E. Eade and T. Drummond, "Scalable monocular slam," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 469–476.

[13] M. Montemerlo and S. Thrun, "Fastslam 2.0," *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pp. 63–90, 2007.

[14] S. Weiss, M. Achtelik, S. Lynen, M. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[15] H. Stewenius, C. Engels, and D. Nistér, "Recent developments on direct relative orientation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 4, pp. 284–294, 2006.

[16] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[17] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] P. J. Huber, *Robust statistics*. Springer, 2011.

[19] 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*. IEEE, 2012, pp. 2815–2821.

[20] L. Kneip, S. Weiss, and R. Siegwart, "Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2235–2241.

[21] J. Lobo and J. Dias, "Inertial sensed ego-motion for 3d vision," *Journal of Robotic Systems*, vol. 21, no. 1, pp. 3–12, 2004.

[22] T. Viéville, E. Clergue, and P. D. S. Facao, "Computation of ego motion using the vertical cue," *Machine Vision and Applications*, vol. 8, no. 1, pp. 41–52, 1995.

[23] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of imu and vision for absolute scale estimation in monocular slam," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 287–299, 2011.

[24] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3923–3929.

[25] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 519–535, 2007.

[26] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[27] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.

[28] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[29] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 834–849.

[30] S. Leutenegger, P. T. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization." 2013.

[31] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[32] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.