Obstacle Detection Using Monocular Camera for Low Flying Unmanned Aerial Vehicle

by

Fan Zhang, **B.Eng.**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
January, 2015

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Obstacle Detection Using Monocular Camera for Low Flying Unmanned Aerial Vehicle

submitted by **Fan Zhang**, **B.Eng.**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

---

Rafik Goubran, Thesis Supervisor

Paul Straznicky, Thesis Supervisor

Roshdy Hafez, Department Chair,
Department of Systems and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
January, 2015

Abstract

This thesis describes the research of an obstacle detection system for a low flying autonomous unmanned aerial vehicle (UAV). The system utilized an extended Kalman filter based simultaneous localization and mapping algorithm which fuses navigation measurements with monocular image sequence to estimate the poses of the UAV and the positions of landmarks.

To test the algorithm with real aerial data, a test flight was conducted to collect data by using a sensors loaded simulated unmanned aerial system(SUAS) towed by a helicopter. The results showed that the algorithm is capable of mapping landmarks ranging more than 1000 meters. Accuracy analysis also showed that SUAS localization and landmark mapping results generally agreed with the ground truth.

To better understand the strength and weakness of the system, and to improve future designs, the algorithm was further analyzed through a series of simulations which simulates oscillatory motion of the UAV, error embedded in camera calibration result, and quantization error from image digitization.

Acknowledgments

I would like to thank my supervisor Professor Rafik A. Goubran and Professor Paul V. Straznicky for the tremendous level of guidance and support. Without them, this research would not have been possible. I would also like to thank Sander Geophysics Limited for carrying out the test flight and providing the raw data which is invaluable to this research. Additionally, I would like to thank the Ontario Centers of Excellence and NSERC (Collaborative Research and Development Grant and Discovery Grant) for their financial support.

At last but not the least, I would like to thank my husband Hao Liu, my parents, and my parents in law for their support over the years of my studies.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronyms	Definition
DEM	digital elevation map
DOF	degree of freedom
EKF	extended Kalman filter
FOV	field of view
GPS	global positioning system
IMU	inertial measurement unit
LSB	least significant bit
OD	obstacle detection
PDF	probability density function
SGL	Sander Geophysics Ltd.
SLAM	simultaneous localization and mapping
SSE	sum of squared error

SUAS	simulated unmanned aerial vehicle
UAV	unmanned aerial vehicle

Chapter 1

Introduction

For any autonomous vehicle, generating an accurate and high precision model of its surrounding environment to indicate hazard features, and knowing its own location in the map is essential for the vehicle to navigate and avoid obstacles autonomously.

In many applications, the mobile robot has an a priori map. The given a priori map may be sufficient for localization purposes, but generally does not have sufficient resolution or up-to-date information for obstacle detection. Ground vehicles need to deal with temporary added road blocks and parked cars. Aerial vehicles need a high resolution map that indicates tall trees, steep hills or electrical towers. In addition, a usable map does not always exist. Without maps and externally referenced pose information, the robot must produce its own map and concurrently localize itself within the map. This problem is referred to as simultaneous localization and mapping (SLAM).

Traditional two-dimensional SLAM algorithms have been well established in the past decade. A SLAM algorithm typically utilises measurements from several types of sensors which can be divided into two groups: those that provide vehicle pose measurements, such as a wheel odometer or GPS; and those that provide landmark bearing and range measurements, such as radar, sonar, laser range finder. In recent

years, optical sensors are actively being incorporated into SLAM algorithms and successfully used in ground vehicle navigation. For aerial vehicles, the experiments are mostly limited to simulation [1] [2] [3] [4], and results from realistic aerial video data are rare.

1.1 Problem Statement

Obstacle detection (OD) has received a lot of research interest in recent years. Various algorithms were developed for ground, under water and aerial vehicles using different sensors such as sonar, radar, LIDAR, and vision. Most OD system focused on only one sensor. Yet, using multiple sensors generally produced better measurements than a single sensor [5]. On most unmanned aerial vehicle (UAV) platforms, many sensors are readily available, such as accelerometers, gyroscope, GPS receiver, altimeter, etc. Fully utilizing these sensors should improve the accuracy and robustness of an OD system, especially in harsh flying conditions.

This thesis focused on developing and testing an obstacle detection system by using a SLAM algorithm as a sensor fusion framework to integrate measurements from various sensors on a typical UAV navigation device. The type of application targeted by this work is a medium size UAV conducting low altitude terrain following flight in a natural environment. The obstacles are static objects on ground; moving objects are not considered. Research presented in this thesis contributed to the project of developing a mid-size UAV to perform geological surveys, carried out by Carleton University in collaboration with Sander Geophysics Ltd. which is an Ottawa based company specializing in high precision geophysical survey. To achieve high resolution data acquisition, the UAV must be able to perform terrain following flight with altitude as low as 10 meters from ground, and with a ground speed ranging from 60

knots (30.87 m/s) to 100 knots (51.44 m/s). The specified rate of climb for the UAV is 400ft of vertical rise per minute (122 meters per minute) [6]. A quick analysis on the UAV specification and aerodynamic behavior reveals the detection requirement of the OD system. Assuming a tree height of 20 meters, which is the average height for oak or pine, and giving approximately 10 meters of clearance, the UAV must be able to detect the threat at 303 meters or further when flying at 60 knots, or 505 meters or further when flying at 100 knots. (Figure 1.1). This analysis indicates that the obstacle detection must be able to map objects up to a thousand meters from the UAV.

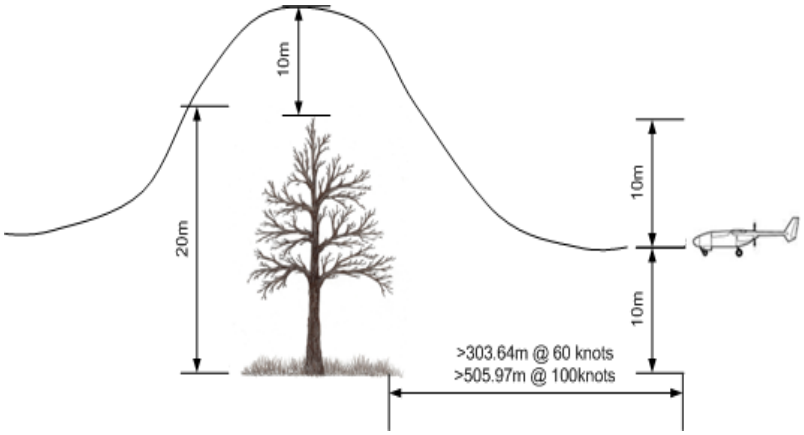

Figure 1.1: Case study for obstacle detection requirement

Although digital terrain maps are generally available for flight path planning and in-flight navigation, they do not have the resolution to indicate all hazardous obstacles such as tall trees, steep hills, or man-made objects. The obstacle detection and avoidance system must be in place to detect discrete threats, and to operate automatically with minimum intervention from the operator.

1.2 Contributions

The thesis first reviews the pros and cons of various sensors, and points out the advantages and disadvantages of using imaging sensors in an OD application. Different types of imaging sensor configurations and sensor calibration methods are also described. Next, the thesis reviews the formulation and properties of a typical extended Kalman filter (EKF) based SLAM algorithm, and discusses the advantages and limitations of an EKF based SLAM algorithm.

The reviews and discussions lead to implementation of an improved EKF SLAM method by fusing multiple sensors with monocular camera video. A camera centric EKF based SLAM algorithm (referred to as CC-EKF-SLAM in the rest of the article) is described in this thesis. The algorithm utilizes an extended Kalman filter to fuse camera motion measurements from inertial and gyroscope sensors and landmark measurements from video of a single wide angle camera. Inverse parameterization was adopted to describe the landmark positions so that distant landmarks can be estimated. Camera centric coordinate system were used to improve the consistency of the framework for large area mapping. The filter can estimate absolute coordinates of landmarks and the poses of the UAV directly. An interpolated map can be generated from the estimates to represent the surrounding environment of the UAV, and the location of the UAV within it.

To test the algorithm under true flying condition, aerial flight data were collected and processed by the CC-EKF-SLAM algorithm. To capture low altitude flight video and navigation data, a simulated unmanned aerial system (SUAS) with various sensors on-board was towed by a helicopter which flew a pre-planned path in the mountain north of Gatineau, Quebec. Sensors installed included 1 CCD camera with 6mm focal length lens, GPS antenna, a UAV navigation module with embedded

accelerometer, gyroscope, GPS receiver, and external fluxgate sensor. The helicopter flew at a speed of 60 knots, and at an altitude of 100 meters above ground, with the SUAS at approximately 70 meters above ground. Two pieces of video and data were processed by the CC-EKF-SLAM algorithm. The result proved that the CC-EKF-SLAM algorithm was capable of mapping landmarks over 1000 meters away. When equipped with a high resolution high dynamic range camera, the system would be able to produce high resolution model of the surrounding environment, and detect static obstacles. The preliminary result of the test flight was published in [7]. This paper was one of the first in the field that successfully applied monocular vision SLAM in large scale aerial OD application. Most researches on UAV OD or SLAM had no real aerial data result, or not tested for distant objects [8], [9], [10], [11].

To further analyze errors seen in the flight test results, a series of simulations were done to thoroughly study the behavior of the algorithm under various circumstances, including

- UAV conducting simple forward motion
- UAV experiencing oscillatory motion on all other 5 degree of freedoms (DOFs)
- error in camera calibration
- quantization error introduced through image digitization

The results of these simulations are valuable to the design of data acquisition hardware for obstacle detection purposes, and to the future improvement of the CC-EKF-SLAM algorithm.

1.3 Organization

The thesis is organized as follows:

- Chapter 2 presents an overview on sensors, and data fusion framework (EKF and SLAM) related to obstacle detection and range measurement.

- Chapter 3 describes the experiment setup for the aerial data collection, camera calibration procedure and result, and ground truth data collection.

- Chapter 4 describes the detailed implementation of the proposed CC-EKF-SLAM algorithm. Data preparation steps used to compare estimated data with the ground truth are also given in this chapter.

- Chapter 5 presents the results of the flight test. Convergence and consistency of the algorithm are discussed. Accuracy analysis by comparing to the ground truth data is also presented.

- Chapter 6 presents the results of error analysis through simulations. Behavior of the CC-EKF-SLAM algorithm was studied through simulating a number of scenarios.

- Chapter 7 gives an overall summary of the results obtained in this research, and recommendations for future research.

Chapter 2

Literature Review

2.1 Sensors for Obstacle Detection

2.1.1 Overview

Many types of sensors can be used for obstacle detection such as ultrasonic sensor, laser range finder, sonar, image sensor, or 3D flash LIDAR [12] [13] [14] [15] [16] [17] [18] [8] [19] [20] [21] [22]. Laser range finders and ultrasonic sensors such as radar and sonar are capable of high accuracy range measurement to centimeter level, but only provide point measurement at a given position and orientation. To acquire a full 3D range map of a scene, 2D mechanical scanning is required, which limits the data acquisition rate of these devices. LIDAR operates in the same manner as a laser range finder, but with the scanning mechanism built in. Although LIDAR is becoming more and more popular on land robots, it is usually expensive, heavy [23], and has a high power requirement when long range measurement is required [24]. These characteristics make it unsuitable for a mid-size economical UAV. Sonar is usually used in indoor or underwater applications, and has a wide beam profile which makes it difficult to identify the origin of return signals and results in a low resolution range

map. 3D flash LIDAR is capable of acquiring 3D range measurements simultaneously by illuminating the entire field of view (FOV) of the camera with a single laser, and capturing the reflected laser with a 2D imaging sensor [21]. However, its high cost has limited its use in commercial applications. In recent years, many researchers have used image sensors as passive range sensors due to their low weight, and low cost. With advances in computer vision technology, image sensors have been successfully used for range mapping and obstacle detection in a number of platforms [25] [26] [27] [7] [28] [29] [30] [18] [31] [32] [33] [34].

2.1.2 Monocular Vision and Binocular Vision

Image sensors are bearing-only sensors, which provide measurement on the direction of the landmarks, but not the range. Other sensors mentioned above, such as radar, LIDAR, are range and bearing sensors. The principle of range measurement for image sensors is through triangulating a common scene point in two or more images captured. There are two types of configuration: monocular, and binocular.

For a binocular camera setup, two cameras are placed apart from each other with their relative geometry (baseline separation and optical axis orientation) fixed and known. The same scene point must be captured by both cameras simultaneously. When the position of a landmark can be accurately found in both images, its disparity is defined by the difference in coordinates of the landmark in the left and right images. Then, the coordinates of the landmark in world can be calculated by using disparity, and the relative geometry of the cameras. Because a binocular camera configuration acquires two images simultaneously, landmark distance can be obtained at time 0. The challenge in binocular range sensing is the correspondence between the two images, i.e., finding the same scene point in left and right images. In addition, because of the limited separation between the cameras, and vibration noise caused by vehicle motion,

binocular range sensing is usually applied in short to medium range measurement.

For a monocular camera setup, only one camera is used with an odometry sensor providing camera displacement measurement between frames. Then a similar principle is applied to triangulate the common landmark in two consecutive frames to measure the distance. If the camera is facing the direction of travel, data association is less difficult as image differences from frame to frame are usually small. Secondly, the baseline separation can be selected according to the targeted object distance by processing every frame, every other frame, or every n frames. This flexibility allows a monocular configuration to be used for longer range measurement. On the other hand, since monocular camera captures 1 frame at a time, the earliest range measurement is not available until two image frames are captured.

2.1.3 Camera Calibration

The camera model relates measurements on the image plane to measurements of the 3D world. Camera calibration is the process of estimating the camera model, which includes projection geometry of a camera and distortion model of the lens. These models define the *intrinsic parameters* of a camera.

Basic Projection Geometry

A 3D point $\boldsymbol{P} = (X, Y, Z)$ and the corresponding point $\boldsymbol{p} = (x, y)$ on the image plane (Figure 2.1), are related by equation 2.1 [35]

$$x = f_x \left(\frac{X}{Z}\right) + c_x, \quad y = f_y \left(\frac{Y}{Z}\right) + c_y \tag{2.1}$$

where (c_x, c_y) is the pixel coordinate at which the optical axis intersects the image plane; f_x and f_y (in pixels) are the product of focal length f (in millimeters) and the

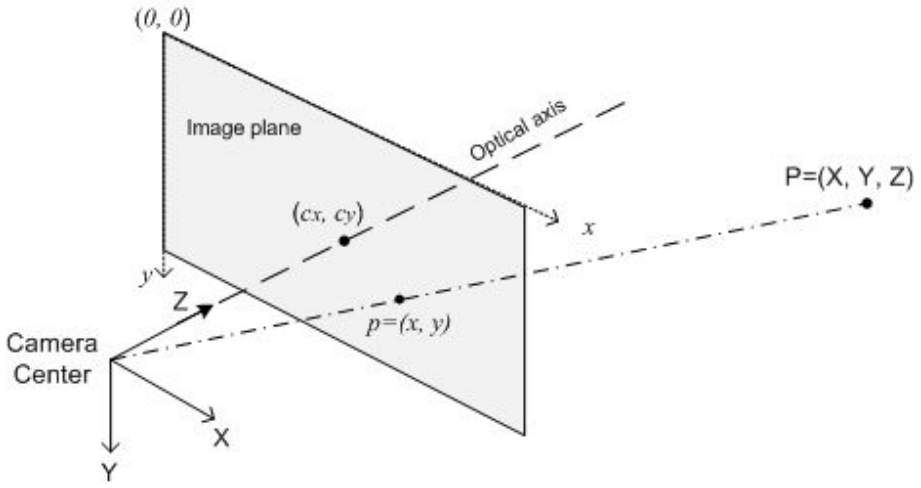

Figure 2.1: Pinhole camera model

world-to-image scaling factor s_x and s_y (in pixels/millimeter) for the X and Y axes. The parameters c_x, c_y, f_x, f_y define the camera intrinsic matrix [35] [36].

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Lens Distortion

There are two main contributors to lens distortion: radial distortion and tangential distortion. Radial distortion arises as a result of the shape of lens. It causes noticeable distortion to the pixels close to the edge of the image, resulting in a "fish-eye" effect. The radial distortion is zero at the center of the image, and increases with the distance to the optical center. It is characterized by the first few terms of a Taylor series expansion [35]

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.3}$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.4}$$

where r is the distance of the landmark from the optical center on the image plane. Tangential distortion arises from misalignment between the image sensor and the lens. It is minimally characterized by two additional parameters p_1 and p_2 [35]

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)] \tag{2.5}$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 x] \tag{2.6}$$

Camera Calibration Algorithm

Camera calibration is a process of finding the intrinsic parameters of a camera. There are two popular camera calibration methods. Tsai's method [37] requires at least 5 feature points in 3D space with known coordinates, and their corresponding coordinates on the image plane. With Tsai's method, only one snapshot of the calibration target is needed. The disadvantage of this method is that the 3D positions of the feature points are difficult to obtain. Zhang [38] proposed a more flexible method that requires at least two different views of a coplanar target, which is usually a black and white checkerboard image. However, to get a more reliable result, 10 views of the checkerboard with different translations and rotations from the camera are desired. The target's coordinates in the world frame need not be known in advance. Instead, Zhang's algorithm only requires the size of the square on the checkerboard. Zhang's method was implemented in OpenCV [35], and was used to calibrate the camera for this work.

2.2 SLAM as a Sensor Fusion Framework

An essential aspect of autonomy for a mobile robot is the capability of obtaining a map and determining its location within it. This problem has been referred to as simultaneous localization and mapping (SLAM). There has been a considerable amount of research done on the SLAM problem in the last two decades, and the theoretical SLAM can now be considered solved. The standard structure of a SLAM problem is a Bayesian form, and explains the evolution of the SLAM process. The most common computational solution is through the use of an extended Kalman filter (EKF).

2.2.1 Recursive Probabilistic Estimation Using Extended Kalman Filter

The Kalman filter [39], published by R. E. Kalman in 1960, is a very powerful recursive data processing algorithm for dynamic stochastic processes. The filter is extensively used in control and navigation applications for its ability to estimate past, present and even future states. It is an attractive candidate for a data fusion framework as it can process all available measurements including previous knowledge of the process, regardless of their precision, to estimate the current value of the state variables. Given a dynamic process that satisfies the assumptions that the Kalman filter is based on, the filter is the optimal algorithm in minimizing the mean squared error of the state variables. This section briefly summarizes the assumptions and formulations of the Kalman filter which is described in detail in [40] [41] [42] [43] [44]. A more intuitive introduction can be found in chapter 1 of [45].

A Kalman filter has three assumptions. 1) *The system model is linear.* The linearity is desired in that the system model is more easily manipulated with engineering

tool. When nonlinearity exists, the typical approach is to linearize the system model at some nominal points. 2) *The noise embedded in system control and measurements is white.* This assumption implies that the noise value is not correlated in time, and has equal power at all frequencies. 3) *The probability density function (PDF) of system and measurement noise is Gaussian.* A Gaussian distribution is fully represented by the first and second order statistic (mean and variance) of a process. Most other PDFs require many higher order of statistics to describe the shape fully. Hence, when the PDF of a noise process is non-Gaussian, the Kalman filter that propagates the first and second order statistics only includes partial information of the PDF.

Kalman Filter Models

The Kalman filter requires two models: a process model and a measurement model. The process model defines a discrete-time controlled process using a linear stochastic difference equation.

$$\text{Process Model: } \boldsymbol{x_k} = \boldsymbol{A x_{k-1}} + \boldsymbol{B \mu_{k-1}} + \boldsymbol{w_{k-1}} \tag{2.7}$$

The $n \times n$ matrix $\boldsymbol{A}$ relates the state variables $\boldsymbol{x_{k-1}}$ in the previous time step $k-1$ to the state variable $\boldsymbol{x_k}$ in the current time step k. The matrix $\boldsymbol{B}$ relates the optional control input $\boldsymbol{\mu}$ to the state variables $\boldsymbol{x}$. Given measurement vector $\boldsymbol{z_k}$ of size $m \times 1$, the measurement model relates the state variables to the measurements by matrix H of size $m \times n$.

$$\text{Measurement Model: } \boldsymbol{z_k} = \boldsymbol{H x_k} + \boldsymbol{v_k} \tag{2.8}$$

The random variables $\boldsymbol{w}$ and $\boldsymbol{v}$ represent the uncertainty or noise of the process model, and the measurement. $\boldsymbol{w}$ and $\boldsymbol{v}$ are assumed to be unrelated to each other, and have Gaussian distribution with covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Kalman Filter Algorithm

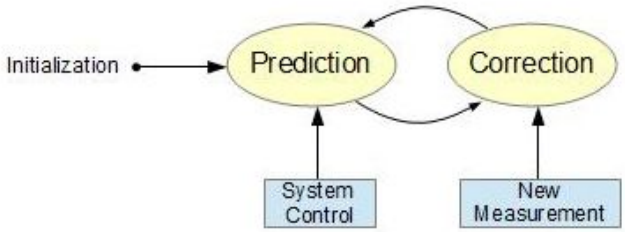

Figure 2.2: Kalman filter operation flow diagram

The Kalman filter operates in prediction and correction cycles after initialization (Figure 2.2). The state vector $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$, $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+}$ contain a priori and a posteriori estimates of the variables of interest at time step k. $\boldsymbol{P}_{\boldsymbol{k}}^{-}$ and $\boldsymbol{P}_{\boldsymbol{k}}^{+}$ are the a priori and a posteriori covariance matrices of the state vector. The equations for prediction and correction cycles are listed in Table 2.1. In prediction, an estimate of the state vector is made based on the known process model. Since there are always unknown factors not being fully described by the process model, the errors almost always increase in the prediction. During correction, a number of steps are carried out to correct the a priori estimate. First, the predicted measurements $\boldsymbol{H}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$, calculated through the measurement model, are compared to the new measurements $\boldsymbol{z}_{\boldsymbol{k}}$. The difference $\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$ is called measurement innovation, or residual. Next, the residuals are weighted by the Kalman gain $\boldsymbol{K}$, and are added to $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$ as correction. The Kalman gain is formulated so that it minimizes the a posteriori error covariance matrix $\boldsymbol{P}_{\boldsymbol{k}}^{+}$.

Table 2.1: Kalman filter operation equations

Prediction	$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}^{+} + \boldsymbol{B}\boldsymbol{\mu}_{\boldsymbol{k-1}}$	(2.9)
	$\boldsymbol{P}_{\boldsymbol{k}}^{-} = \boldsymbol{A}\boldsymbol{P}_{\boldsymbol{k-1}}^{+}\boldsymbol{A}^{\boldsymbol{T}} + \boldsymbol{Q}$	(2.10)
Correction	$\boldsymbol{K}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-1}$	(2.11)
	$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+} = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K}_{\boldsymbol{k}}(\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-})$	(2.12)
	$\boldsymbol{P}_{\boldsymbol{k}}^{+} = (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{H})\boldsymbol{P}_{\boldsymbol{k}}^{-}$	(2.13)

Extended Kalman Filter

When a discrete-time controlled process, or its relationship with the measurements is non-linear, a Kalman filter must be linearized about the estimated trajectory. The filter is referred to as an extended Kalman filter or EKF. A process with state vector $\boldsymbol{x}$ and measurement $\boldsymbol{z}$ that is governed by a non-linear stochastic difference equation has process and measurement model

$$\boldsymbol{x}_{\boldsymbol{k}} = f(\boldsymbol{x}_{\boldsymbol{k-1}}, \boldsymbol{\mu}_{\boldsymbol{k-1}}, \boldsymbol{w}_{\boldsymbol{k-1}}), \tag{2.14}$$

$$\boldsymbol{z}_{\boldsymbol{k}} = h(\boldsymbol{x}_{\boldsymbol{k}}, \boldsymbol{v}_{\boldsymbol{k}}), \tag{2.15}$$

where the random variables $\boldsymbol{w}_{\boldsymbol{k}}$ and $\boldsymbol{v}_{\boldsymbol{k}}$ represent the process and measurement noise with covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$. The extended Kalman filter operation equations are given in Table 2.2,

Table 2.2: Extended Kalman filter operation equations

Prediction	$\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}^+, \boldsymbol{\mu_{k-1}}, \boldsymbol{0})$	(2.16)
	$\boldsymbol{P_k^-} = \boldsymbol{A_k P_{k-1}^+ A_k^T} + \boldsymbol{W_k Q_{k-1} W_k^T}$	(2.17)
Correction	$\boldsymbol{K_k} = \boldsymbol{P_k^- H_k^T}(\boldsymbol{H_k P_k^- H_k^T} + \boldsymbol{V_k R_k V_k^T})^{-1}$	(2.18)
	$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^+ = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^- + \boldsymbol{K_k}(\boldsymbol{z_k} - h(\hat{\boldsymbol{x}}_{\boldsymbol{k}}^-, \boldsymbol{0}))$	(2.19)
	$\boldsymbol{P_k^+} = (\boldsymbol{I} - \boldsymbol{K_k H})\boldsymbol{P_k^-}$	(2.20)

where (subscript k is omitted)

- $\boldsymbol{A}$ is the Jacobian matrix of partial derivatives of f with respect to $\boldsymbol{x}$,

$$A_{[i,j]} = \frac{\partial f_i(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}^+, \boldsymbol{\mu_{k-1}}, 0)}{\partial x_j} \tag{2.21}$$

- $\boldsymbol{W}$ is the Jacobian matrix of partial derivatives of f with respect to $\boldsymbol{w}$,

$$W_{[i,j]} = \frac{\partial f_i(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}^+, \boldsymbol{\mu_{k-1}}, 0)}{\partial w_j} \tag{2.22}$$

- $\boldsymbol{H}$ is the Jacobian matrix of partial derivatives of h with respect to $\boldsymbol{x}$,

$$H_{[i,j]} = \frac{\partial h_i(\hat{\boldsymbol{x}}_{\boldsymbol{k}}^-, 0)}{\partial x_j} \tag{2.23}$$

- $\boldsymbol{V}$ is the Jacobian matrix of partial derivatives of h with respect to $\boldsymbol{v}$,

$$V_{[i,j]} = \frac{\partial h_i(\hat{\boldsymbol{x}}_{\boldsymbol{k}}^-, 0)}{\partial v_j} \tag{2.24}$$

Note that when $\boldsymbol{w}$ and $\boldsymbol{v}$ directly describe the noise of state vector and measurement, the Table 2.2 is the same as Table 2.1.

Tuning

The tuning of a Kalman filter can be achieved by adjusting the noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The measurement noise covariance $\boldsymbol{R}$ is usually easy to estimate, i.e., by taking some offline measurements to compute the covariance. On the other hand, process noise covariance $\boldsymbol{Q}$ is more difficult. One common way is to inject enough uncertainty into the process noise, and rely on reliable measurements.

2.2.2 SLAM with Extended Kalman Filter

A simultaneous localization and mapping (SLAM) algorithm allows a robot to build a map of its surrounding, and simultaneously determining its location within the map. The mapping is achieved by estimating the position of a number of landmarks, which are arbitrary points in the environment that are detected and tracked by the algorithm. When sufficient amount of landmarks can be established, and evenly distributed among the scene, a high resolution map can be obtained.

The structure of the SLAM problem has evolved to a standard Bayesian form. The EKF implementation of a SLAM problem is reviewed in this section. A more complete tutorial to the SLAM problem is given in [46] [47].

General EKF Model for SLAM

Previous research showed that there is a high degree of correlation between the estimates of the landmarks [48] [49]. This correlation exists because of the common error in the estimated vehicle poses [50]. The implication of these works is that a consistent full solution to the SLAM problem requires a joint state composed of the vehicle poses

and all landmark positions [46]. When the vehicle poses and landmark positions are formulated as one single estimation problem, the result is convergent. The correlation between landmarks plays an important role in the quality of the solution. The more the correlation grows, the better the solution becomes [51] [52] [53] [54].

To formulate a SLAM problem, the following variables are defined at a given time step k

- $\boldsymbol{x_k}$: a vector that describes the vehicle position and orientation.
- $\boldsymbol{u_k}$: the control vector applied at time $k-1$ to drive the vehicle to $\boldsymbol{x_k}$ at time step k
- $\boldsymbol{p_i}$: a vector that describes the location of the i^{th} landmark. All landmarks locations are assumed to be time invariant.
- $\boldsymbol{z_{i,k}}$: observation of the location of the i^{th} landmark taken from the vehicle's location at time step k

A complete state vector contains

$$\begin{bmatrix} \hat{\boldsymbol{x}}_{\boldsymbol{k}} \\ \\ \hat{\boldsymbol{p}}_{\boldsymbol{k}} \end{bmatrix} \tag{2.25}$$

and the state covariance matrix contains

$$\begin{bmatrix} \boldsymbol{P_{xx}} & \boldsymbol{P_{xp}} \\ \\ \boldsymbol{P_{px}} & \boldsymbol{P_{pp}} \end{bmatrix} \tag{2.26}$$

The prediction and correction cycles can then be carried out following the standard EKF algorithm. The landmarks are generally not updated at the prediction (i.e. only

$\boldsymbol{x_k}$ and $\boldsymbol{P_{xx}}$ are updated) unless they are moving.

Properties of SLAM

Dissanayake [54] reached three important convergence properties of Kalman filter based SLAM, namely that:

1. *the determinant of any sub-matrix of the map covariance matrix decreases monotonically as observations are successively made;*

2. *in the limit as the number of observations increases, the landmark estimates become fully correlated;*

3. *in the limit the covariance associated with any single landmark location estimate reaches a lower bound determined only by the initial covariance in the vehicle location estimate at the time of the first sighting of the first landmark.*

These results show that a complete knowledge of the cross-correlation between landmark estimates is critical in maintaining the structure of a SLAM problem. The errors in the estimates of landmarks become more and more correlated as the vehicle ventures through the unknown environment. The errors eventually become fully correlated which means that given the location of one landmark, the location of any other landmark can be determined with absolute certainty. As the map converges, the errors in the estimates of landmarks reduce to a lower bound determined by the errors of vehicle poses when the first observation was made.

The results above only refer to the evolution of covariance matrices computed by a linear model. In reality, most SLAM problems are nonlinear, and the computed covariance does not match the true estimation error. This leads to a SLAM consistency issue.

Linearization Error and Consistency

Many researchers reported filter divergence due to linearization errors [55] [56] [57] [58] [59]. As defined in [60], a state estimator is consistent if the estimation errors (i) are zero-mean, and (ii) have covariance matrix smaller or equal to the one calculated by the filter. When covariance reduces to a value smaller than the actual error, over-optimistic estimation occurs and the filter can no longer update the estimates effectively.

Huang investigated the properties and consistency of the nonlinear two-dimensional EKF SLAM problem. He derived and proved a number of theorems applicable to EKF SLAM. The conclusions are [61]:

- *Most of the convergence properties in [54] are still true for the nonlinear case provided that the Jacobians used in the EKF equations are evaluated at the true states.*

- *The main reasons for inconsistency in EKF SLAM are*

    1. *the violation of some fundamental constraints governing the relationship between various Jacobians when they are evaluated at the estimated location, instead of the true location*

    2. *the use of relative location measurements from robot to landmarks to update the absolute robot and landmark location estimates in world coordinate.*

- *The robot orientation uncertainty plays an important role in both the EKF SLAM convergence and the possible inconsistency.*

2.2.3 SLAM for Building Large Scale Maps

The consistency issue of EKF SLAM limits its application in mapping a large area. In response, a number of methods were proposed.

Robot Centric Coordinate System

Robot-centric mapping was proposed by Castellanos et al. [59] and was adopted in [62] to create a large scale map on land. This approach uses a reference frame attached to the robot as the base frame. All geometric measurements and estimations were referred to the robot frame. This method was shown to improve the consistency of the EKF based solution to SLAM problem, but using local map joining to produce a large map gives better result.

Sub-map Methods

The sub-map method is another way to tackle the large scale map problem. The sub-maps can be either globally referenced or relatively referenced. Global sub-maps estimate the location of the sub-map referenced to a common base frame [63] [64]. However, as the sub-map frames are referred to a common base frame, a global sub-map does not improve the consistency issue caused by the robot pose uncertainty [47]. A relatively referenced sub-map records its location referenced to the neighboring sub-maps [17] [16]. A global map is then obtained through vector summation. The relatively referenced sub-map is able to alleviate linearization problems in a large scale map, but does not converge at a global level without an independent sub-map structure. This problem can be solved by using the Atlas framework or network coupled feature maps [65] [66].

2.3 Inverse Depth Parameterization for Distant Object

The standard way of describing a landmark position is through the Euclidean XYZ parameterization. In an outdoor range estimation problem, the algorithm must deal with landmarks located at near infinity, which give poor linearity when being represented in a traditional Euclidean system. Furthermore, these landmarks cannot be added to the filter until their locations have been determined with good certainty. On the other hand, the benefit of including landmarks at infinity is that they contribute in estimating the camera rotational motion even though they offer little information on camera translational motion. Inverse depth parameterization overcomes this problem by representing range d in its inverse form $\rho = 1/d$. It also allows for initializing long distance landmark into the EKF framework before the landmark can be safely triangulated. The inverse depth concept is widely used in computer vision for its relation with the image disparity in stereo vision, or the optical flow vectors in motion stereo [67] [68] [69]. Research from Aidala et al. [70] shows that modified polar coordinates leads to a stable extended Kalman filter in bearing-only target motion analysis. Civera et al. [19] introduced an inverse depth parameterization in polar coordinates. The inverse depth of a landmark is referenced to the position at which the landmark was first observed. This method results in measurement equations with high degree of linearity.

Chapter 3

Experiments with Real Aerial Data

3.1 Equipment Setup and Flight Data Collection

The biggest contribution of the project is that the proposed algorithm was tested and proved feasible with real flight data collected by the author. The aerial video and navigation data were collected through a survey flight with the support of Sander Geophysics Ltd. A main purpose of the test flight was to obtain aerial video with the camera close to the ground to mimic the scenario of a low flying UAV. This is difficult to achieve with any manned fixed wing aircraft since terrain following flight at low altitude is very dangerous. Therefore, a helicopter was used to conduct the survey flight to achieve the terrain following action. To get sensors close to the ground and to allow for flexible sensor mounting, a simulated unmanned aircraft system (SUAS) was used to carry all sensors. The SUAS was towed by a helicopter via a tow rope of 33 meters length (Figure 3.1). Yet, sufficient clearance must be established between the SUAS and the vegetation to prevent the SUAS from being caught by tree branches. As a result, the helicopter flew a planned path at approximately 100 meters above ground, and the SUAS was at approximately 70 meters above ground.

Figure 3.1: SUAS towed by a helicopter

Sensors mounted on the SUAS (Figure 3.2) included one wide angle CCD camera with 480x720 resolution and 6mm focal length lens capturing monocular image sequence at 30 frames per second, a pair of narrow angle CCD cameras for binocular image sequences, a GPS antenna, a three-axes fluxgate sensor, and Athena GS-111m. The Athena GS-111m is a flight control INS/GPS navigation unit [71] that captures, estimates and outputs 16 channels of data, such as velocity, acceleration, rotation, altitude, GPS coordinates, at a sample rate of up to 100Hz per channel. Analog video and navigation data were sent to the helicopter via three BNC cables and one data cable for recording. Installed in the helicopter were two SGL data acquisition systems, named CDAC (Figure 3.4). The CDAC on top recorded video from the monocular camera, navigation data from the Athena GS-111m, as well as data from sensors installed on the helicopter, including GPS, radar altimeter, laser altimeter, air pressure, temperature, humidity, etc. The CDAC at the bottom recorded video from the binocular cameras. Videos from the three cameras were digitized using three Parvus MPEG4 video encoders installed in the CDACs, and time-stamped with GPS second on the image screen (Figure 3.5) for post-flight synchronization with the navigation data. Because the test flight recorded data from all available sensors on board the SUAS and the helicopter, these recordings would still be useful should any future research require more sensor data.

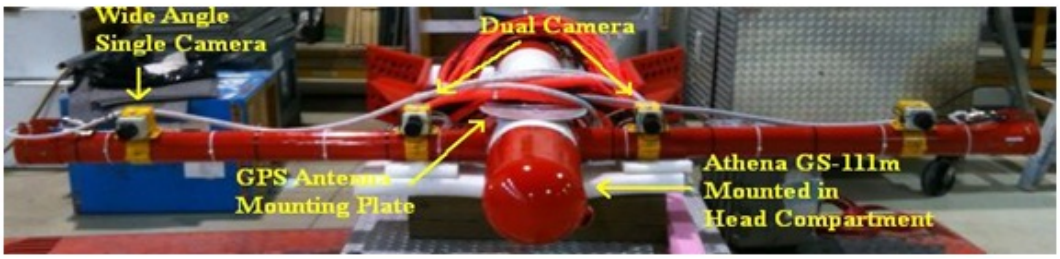

Figure 3.2: Sensors mounting location on SUAS

Figure 3.3: Sensors mounted on SUAS. Top left: Athena GS-111m, top right: GPS antenna, bottom: monocular CCD camera

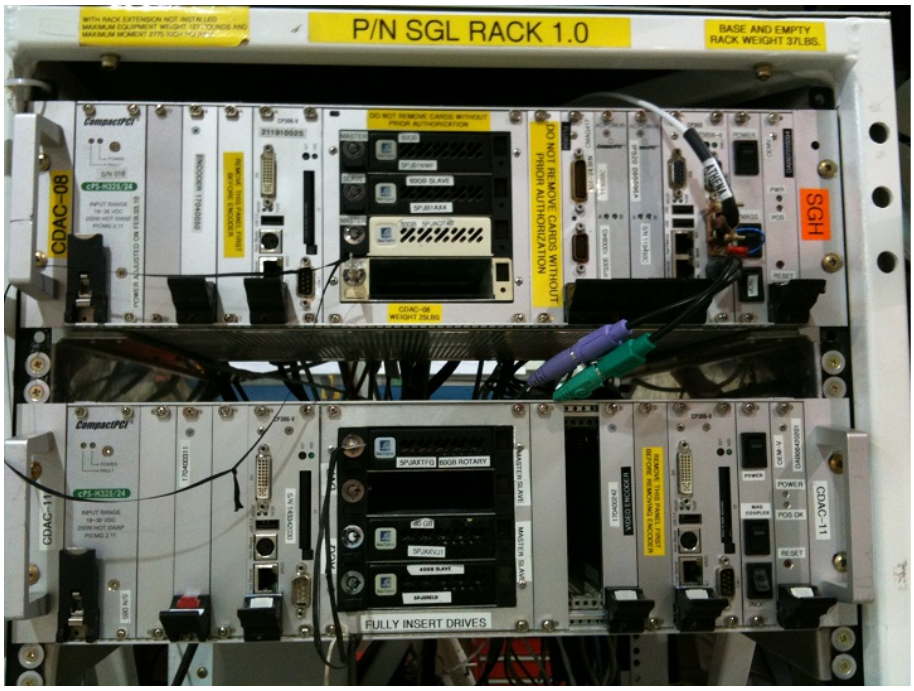

Figure 3.4: Compact PCI data acquisition system (CDAC)

Figure 3.5: Image from monocular camera with GPS second timestamp

3.2 Camera Calibration

Camera calibration decodes the relation between image pixels and the actual 3D world. The intrinsic parameters could be affected by a number of environmental conditions, such as temperature, and humidity. To extract the camera parameters at a camera condition as close as possible to the one during test flight, camera calibration was performed right after the SUAS returned to the SGL hanger. The camera calibration was done by taking a video of a checkerboard pattern with various translations and orientations from the camera. A total of 20 views of the calibration target were chosen from the video, and fed to the calibration algorithm. A few examples are shown in Figure 3.6.

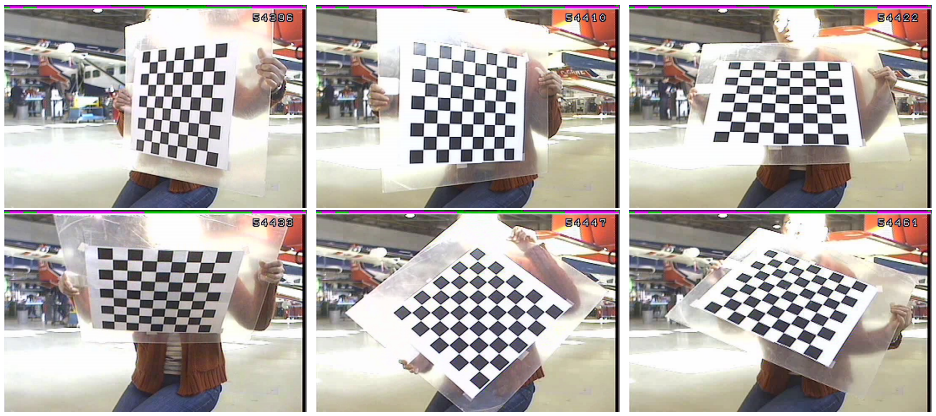

Figure 3.6: A subset of camera calibration input images

The program "calibration.exe" was used to calibrate the camera. This program came with the OpenCV installation. The digitized images have a resolution of 720 pixels in width and 480 pixels in height. Table 3.1 below lists the calibration results.

Table 3.1: Camera calibration result

Parameter	Result
f_x	887.6 pixels
f_y	805.7 pixels
c_x	381.8 pixels
c_y	293.7 pixels
k_1	-0.102
k_2	-0.535
p_1	1.15e-003
p_2	8.40e-003

3.3 Ground Truth Data Collection

The localization ground truth was obtained through the flight control unit GS-111m on board the SUAS. The unit recorded the SUAS position in GPS longitude and latitude coordinates. Orientation was obtained from the roll pitch and heading measurements. Roll and pitch accuracy has 0.1° and 0.1° standard deviation. Heading accuracy can achieve 0.5° [71].

Landmark position ground truth came from a digital elevation map (DEM) downloaded from the CGIAR-CSI website [72]. The DEM contains longitude, latitude and sea level elevation of the terrain with a resolution of approximately 90 meters by 90 meters. The reported vertical error of the DEM is less than 16 meters.

Chapter 4

Description of CC-EKF-SLAM

The proposed algorithm: Camera Centric EKF SLAM (CC-EKF-SLAM), is described in this chapter. The inverse depth parameterization is described first, then followed by the EKF models and procedures. The overall flow chart of the algorithm, and the run time of the current implementation are also given. The preliminary test result on real flight data was published in [7].

The algorithm was implemented in Python programming language [73]. An open source machine vision library OpenCV [74] was utilized to perform machine vision tasks such as feature extraction and tracking. The Shi-Tomasi corner detector [75] was used to extract corner features from the image. The visual tracking is accomplished by the pyramid implementation of the Lucas-Kanade optical flow method [76], referred to as pyramid LK in the rest of the thesis.

4.1 Camera Centric Inverse Depth Parameterization

A 3D scene point $\boldsymbol{p}_i^C$ can be defined by 6 parameters with i representing the landmark number. The superscript C indicates that the parameters are represented in the

camera reference frame.

$$\boldsymbol{p_i^C} = \begin{bmatrix} x_i^C & y_i^C & z_i^C & \rho_i & \varphi_i^C & \theta_i^C \end{bmatrix} \tag{4.1}$$

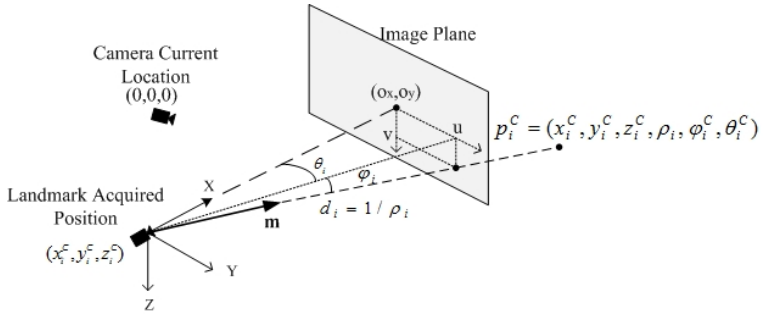

Figure 4.1: Inverse depth parameterization

Figure 4.1 shows the landmark parameters definition in inverse depth parameterization. The first three parameters $[x_i^C, y_i^C, z_i^C]$ represent the initialization point where the landmark is first observed. $\rho_i = 1/d_i$ is the inverse distance from the initialization position to the landmark. The elevation-azimuth pair $[\varphi_i^C, \theta_i^C]$ encodes a unit vector pointing from the initialization point to the landmark. The vector is given by

$$\vec{\boldsymbol{m}}(\varphi_i^C, \theta_i^C) = \begin{bmatrix} \cos\varphi_i^C \cos\theta_i^C \\ \cos\varphi_i^C \sin\theta_i^C \\ \sin\varphi_i^C \end{bmatrix} \tag{4.2}$$

Given the a vector $[h_x, h_y, h_z]$, one can also find the elevation-azimuth angles $[\varphi, \theta]$ by

$$\varphi = arctan\left(\frac{h_z}{\sqrt{h_x^2 + h_y^2}}\right) \tag{4.3}$$

$$\theta = arctan\left(\frac{h_y}{h_x}\right) \tag{4.4}$$

4.2 Modeling the System with Extended Kalman Filter

4.2.1 Full State Vector

The EKF full state vector is defined as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{OX_W^C} & \boldsymbol{c^C} & \boldsymbol{r^C} & \boldsymbol{p_1^C} & \boldsymbol{p_2^C} & \dots \end{bmatrix}^T \tag{4.5}$$

where $\boldsymbol{OX_W^C} = \begin{bmatrix} O_x^C & O_y^C & O_z^C & W_x^C & W_y^C & W_z^C \end{bmatrix}^T$ contains translation parameters $\boldsymbol{O_{x,y,z}^C}$ and orientation parameters $\boldsymbol{W_{x,y,z}^C}$ that represent the world frame position and orientation referenced to the camera frame. $\boldsymbol{c^C}$ and $\boldsymbol{r^C}$ represent the camera translation and rotation motion frame by frame. $\boldsymbol{p_i^C}$ contains the i^{th} landmark parameters described in the previous section.

4.2.2 Prediction

For a prediction step at time k, the world frame parameter $\boldsymbol{OX_W^C}$ and landmarks parameters $\boldsymbol{p_i^C}$ are kept unchanged from time $k-1$. The camera parameters are updated using the new inertial measurements: velocity $\boldsymbol{v^C}$, acceleration $\boldsymbol{a^C}$, and rate of change $\boldsymbol{w^C}$ in roll, pitch, and heading. The full state vector at time k is

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = \begin{bmatrix} \boldsymbol{OX_{W,k-1}^C} & \boldsymbol{c_{measured}^C} & \boldsymbol{r_{measured}^C} & \boldsymbol{p_{1,k-1}^C} & \boldsymbol{p_{2,k-1}^C} & \cdots & \boldsymbol{p_{n,k-1}^C} \end{bmatrix}^T \tag{4.6}$$

where

$$\boldsymbol{c}^{\boldsymbol{C}}_{\boldsymbol{measured}} = \boldsymbol{v}^{\boldsymbol{C}}_{\boldsymbol{measured}}\Delta t + \frac{1}{2}\boldsymbol{a}^{\boldsymbol{C}}_{\boldsymbol{measured}}\Delta t^2 \tag{4.7}$$

$$\boldsymbol{r}^{\boldsymbol{C}}_{\boldsymbol{measured}} = \boldsymbol{r}^{\boldsymbol{C}}_{\boldsymbol{k-1}} + \boldsymbol{w}^{\boldsymbol{C}}_{\boldsymbol{measured}} \tag{4.8}$$

and Δt is the time elapsed from frame to frame.

4.2.3 Measurement Model

Each observed landmark is related to the camera motion through the measurement model (Figure 4.2). This relationship enables a correction on the camera motion and landmark parameters based on the landmark locations observed in the image.

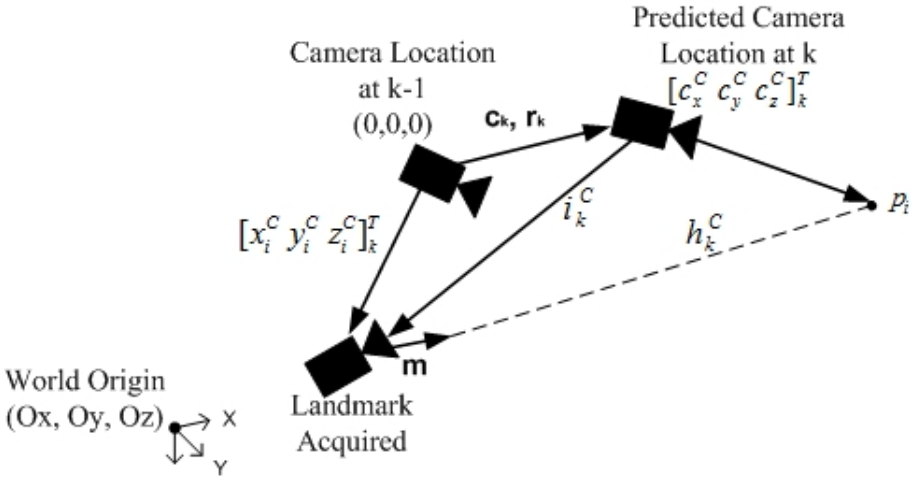

Figure 4.2: Measurement model

For a landmark $\boldsymbol{p}^{\boldsymbol{C}}_{\boldsymbol{i}}$, the vector $\boldsymbol{i}^{\boldsymbol{C}}$ pointing from the predicted camera location to the landmark initialization position is

$$\boldsymbol{i_k^C} = \begin{bmatrix} x_i^C \\ y_i^C \\ z_i^C \end{bmatrix}_k - \begin{bmatrix} c_x^C \\ c_y^C \\ c_z^C \end{bmatrix}_k \tag{4.9}$$

The normalized vector pointing from the predicted camera position to the landmark at time k is then

$$\boldsymbol{h_k^C} = \boldsymbol{Q}^{-1}\left(\boldsymbol{r_k^C}\right)\left(\rho_k \boldsymbol{i_k^C} + \boldsymbol{m}\left(\varphi_k^C, \theta_k^C\right)\right) \tag{4.10}$$

where $\boldsymbol{Q}^{-1}(\boldsymbol{r_k^C})$ is the inverse rotation matrix from the camera frame at time $k-1$ to camera frame at time k. From vector $\boldsymbol{h_k^C}$, the landmark location on image plane can be found by

$$\boldsymbol{h_k^U} = \begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} \frac{f_x h_{y,k}^C}{h_{x,k}^C} \\ \frac{f_y h_{z,k}^C}{h_{x,k}^C} \end{bmatrix} \tag{4.11}$$

where f_x and f_y is the scaling factor of the projection obtained through camera calibration.

Since the measurement model is non-linear, the equation must be linearized to calculate the Kalman gain K. The detailed formulation for linearization is given in Appendix C.1.

4.2.4 Composition

The corrected state vector has all its parameters defined in camera frame at time $k-1$. To keep the reference frame up-to-date, a coordinate transformation is needed

to transform all parameters, and the error matrix into camera frame at step k so that the next cycle of tracking can be continued. This step is referred to as the composition. To differentiate the camera frame, a subscript k and $k-1$ are added to the reference frame indicator C.

World reference frame parameters are transformed using the equation

$$\begin{bmatrix} O_x^{C_k} \\ O_y^{C_k} \\ O_z^{C_k} \end{bmatrix}_k = \boldsymbol{Q}^{-1}(\boldsymbol{r_k^{C_{k-1}}}) \left(\begin{bmatrix} O_x^{C_{k-1}} \\ O_y^{C_{k-1}} \\ O_z^{C_{k-1}} \end{bmatrix}_k - \begin{bmatrix} c_x^{C_{k-1}} \\ c_y^{C_{k-1}} \\ c_z^{C_{k-1}} \end{bmatrix}_k \right) \tag{4.12}$$

and

$$\begin{bmatrix} W_x^{C_k} \\ W_y^{C_k} \\ W_z^{C_k} \end{bmatrix}_k = \begin{bmatrix} W_x^{C_{k-1}} \\ W_y^{C_{k-1}} \\ W_z^{C_{k-1}} \end{bmatrix}_k - \boldsymbol{r_k^{C_{k-1}}} \tag{4.13}$$

Landmark parameters are related to the previous reference frame by

$$\begin{bmatrix} x_i^{C_k} \\ y_i^{C_k} \\ z_i^{C_k} \end{bmatrix}_k = \boldsymbol{Q}^{-1}(\boldsymbol{r_k^{C_{k-1}}}) \left(\begin{bmatrix} x_i^{C_{k-1}} \\ y_i^{C_{k-1}} \\ z_i^{C_{k-1}} \end{bmatrix}_k - \begin{bmatrix} c_x^{C_{k-1}} \\ c_y^{C_{k-1}} \\ c_z^{C_{k-1}} \end{bmatrix}_k \right) \tag{4.14}$$

and

$$\begin{bmatrix} \rho_i \\ \varphi_i^{C_k} \\ \theta_i^{C_k} \end{bmatrix}_k = \begin{bmatrix} \rho_i \\ \boldsymbol{m}^{-1}\left(\boldsymbol{Q}^{-1}(\boldsymbol{r_k^{C_{k-1}}}) \boldsymbol{m}(\varphi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}) \right) \end{bmatrix} \tag{4.15}$$

where ρ_i is a scalar that doesn't require transformation. $\boldsymbol{m}^{-1}$ represents the operator that converts a unit vector back to $[\varphi, \theta]$ angles as defined in Equations (4.3) and (4.4).

The covariance matrix is also affected by this transformation. The new covariance matrix is related to the old one by

$$\boldsymbol{P}_{\boldsymbol{k}}^{\boldsymbol{C_k}} = \boldsymbol{J}_{\boldsymbol{C_{k-1}} \rightarrow \boldsymbol{C_k}} \boldsymbol{P}_{\boldsymbol{k}}^{\boldsymbol{C_{k-1}}} \boldsymbol{J}_{\boldsymbol{C_{k-1}} \rightarrow \boldsymbol{C_k}}^{\boldsymbol{T}} \tag{4.16}$$

where $\boldsymbol{J}_{\boldsymbol{C_{k-1}} \rightarrow \boldsymbol{C_k}}$ is the Jacobian matrix of the composition equations. Detail derivation of the Jacobian matrix is given in Appendix C.2

4.2.5 Filter Initialization

State Vector

The state vector is initialized at the first frame. The world frame position and orientation, camera motions, and the landmark initialization points are all initialized to zero, with variance equal to the smallest machine number.

$$\boldsymbol{OX}_{\boldsymbol{W}}^{\boldsymbol{C}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.17}$$

$$\boldsymbol{c}^{\boldsymbol{C}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.18}$$

$$\boldsymbol{r}^{\boldsymbol{C}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.19}$$

$$\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{C}} = \begin{bmatrix} 0 & 0 & 0 & \rho_i & \varphi_i^C & \theta_i^C \end{bmatrix}^T \tag{4.20}$$

The inverse distance ρ of all landmarks are initialized to 0.01 ($d = 100meters$). The landmark elevation-azimuth angles $[\varphi_i^C, \theta_i^C]$ are extracted from landmark coordinates in the image plane. A vector pointing from camera optical center to a landmark can be defined by

$$\boldsymbol{h^C} = \begin{bmatrix} h_x^C \\ h_y^C \\ h_z^C \end{bmatrix} = \begin{bmatrix} 1 \\ (u - c_x)/f_x \\ (v - c_y)/f_y \end{bmatrix} \tag{4.21}$$

where $[u, v]$ are the landmark coordinates in the image, $[f_x, f_y]$ are the scaling factors of the projection from the scene to the image plane, $[c_x, c_y]$ are the coordinates at which the optical axis intersects the image plane. The elevation-azimuth angles $[\varphi_i^C, \theta_i^C]$ can be directly calculated from $\boldsymbol{h^C}$ using Equations (4.3) and (4.4)

State Covariance Matrix

As the world reference frame is defined by the camera reference frame at time 0, it allows the filter to be initialized with minimum variance, which helps to reduce the lower bound of the filter error according to the EKF SLAM properties. The initial covariance matrix for the world reference frame position and orientation, and the camera motion is

$$\boldsymbol{P} = \boldsymbol{I_{12\times 12}} \cdot \epsilon \tag{4.22}$$

where ϵ is the least significant bit (LSB) of a computer.

The covariance of landmarks are added one by one as there is correlation between them. For every new landmark added, the new covariance matrix becomes

$$\boldsymbol{P_{new}} = \boldsymbol{J} \begin{bmatrix} \boldsymbol{P_{old}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} \boldsymbol{J}^T \tag{4.23}$$

where $\boldsymbol{P_{old}}$ is the covariance matrix of the existing state vector. The initial $\boldsymbol{P_{old}}$ before the addition of the first landmark is Equation (4.22). Matrix $\boldsymbol{R}$ is given by.

$$\boldsymbol{R} = \begin{bmatrix} \sigma_{x_i^C} & & & & & \\ & \sigma_{y_i^C} & & 0 & & \\ & & \sigma_{z_i^C} & & & \\ & & & \sigma_\rho & & \\ & 0 & & & \sigma_{image} & \\ & & & & & \sigma_{image} \end{bmatrix} = \begin{bmatrix} \epsilon & & & & & \\ & \epsilon & & & 0 & \\ & & \epsilon & & & \\ & & & 0.01 & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{bmatrix} \tag{4.24}$$

where $[\sigma_{x_i}^C, \sigma_{y_i}^C, \sigma_{z_i}^C]$ is the uncertainty of the camera optical center position, initialized to ϵ. σ_{image} is the variance of the landmark coordinate on the image plane, set to 1 pixel. σ_ρ is the uncertainty of the inverse distance. Because the filter mainly deals with distant landmark, σ_ρ is initialized to 0.01 to cover distances from 50 meters to infinity.

$\boldsymbol{J}$ in Equation (4.23) is the Jacobian matrix for the landmark initialization equations given in Equations (4.17)-(4.20), (4.3), and (4.4). Detailed formulation of the Jacobian matrix is given in Appendix B.

4.2.6 CC-EKF-SLAM Flow Chart and Run Time

The flow chart of the entire algorithm is shown in Figure 4.3. The algorithm initialize the EKF by detecting 40 corner features from the 1^{st} image frame, and adding the associated landmarks into the EKF state vector. Starting from the 2^{nd} image frame, CC-EKF-SLAM operates on prediction-correction-composition cycles. The prediction is done as described in Section 4.2.2. At the same time, a prediction is also made on the landmark coordinates on the image plane according to the predicted camera motion. In correction process, CC-EKF-SLAM feed the predicted landmark coordinates to the pyramid LK algorithm so that the visual tracking algorithm can start the search for matching pattern at the predicted coordinates. The benefit is shorter search time, and higher accuracy when image illumination level changes. Then, correction to the state vector is made by taking the landmark coordinates returned by the pyramid LK as measurements.

To shorten processing time for each iteration, landmarks that move out of the camera's field of view (FOV) are removed from the filter state vector and covariance matrix. The removal is done by deleting the parameters from the state vector, and the related rows and columns of the state covariance matrix. All tracked landmarks are recorded into a database. The deleted landmarks have their parameters updated based on the camera motion without any further Kalman corrections. Because the deleted landmarks remain in database and are continuously updated, they can be re-injected into the state vector if they re-enter the FOV of the camera.

While some landmarks are deleted, new landmarks are added. The landmark addition occurs after the composition step of each iteration, and follows the same procedure as the filter initialization described in Section 4.2.5.

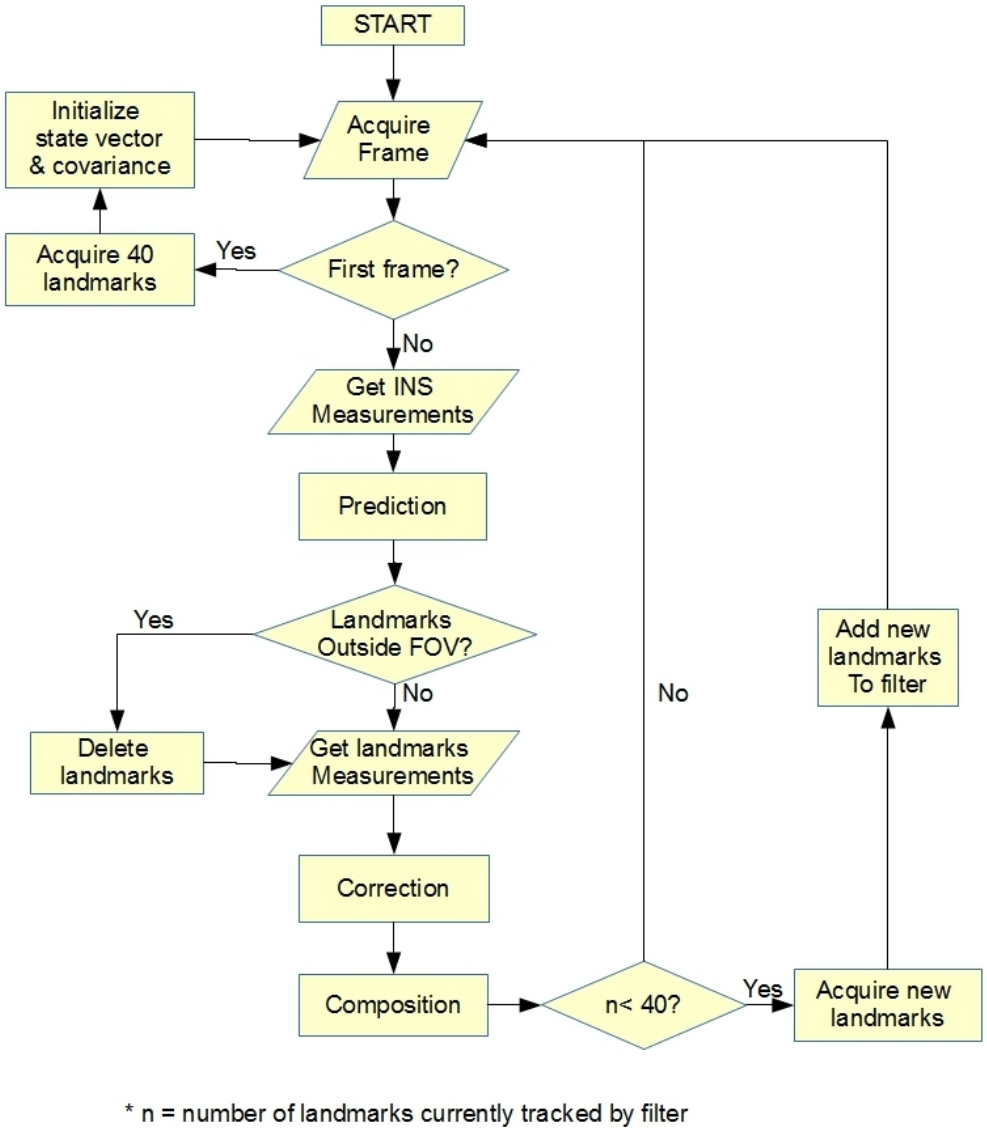

Figure 4.3: Algorithm flow chart

The prototype of non-optimized CC-EKF-SLAM was run on a laptop with Windows 7 operating system installed. A list of hardware and software configuration for the laptop is listed in table 4.1, and the runtime of key component of CC-EKF-SLAM is listed in table 4.2.

Table 4.1: Hardware and software configuration for CC-EKF-SLAM prototyping

Hardware Configuration	
CPU	Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz
RAM	6.00GB
Software Configuration	
Operation System	Windows 7 Home Premium Service Pack 1 64-bit
Python Distribution	Python(x,y) 2.7.3.1 (32-bit)

Table 4.2: Typical runtime for CC-EKF-SLAM

Operation	Time Elapsed
Visual feature extraction	24.99 ms
EKF initialization	23.00 ms
EKF prediction	9.99 ms
EKF correction	19.99 ms
Composition	28.99 ms

4.3 Additional Procedures for Data Analysis

To compare the estimated SUAS poses and landmark positions to the ground truth, all data must be referenced to the same reference frame. For ease of viewing, the reference frame used for data analysis is the world frame defined by the camera frame at time 0.

The longitude and latitude of the SUAS position were first converted to UTM using the WGS84 world geodetic system [77]. Many software packages are readily available to do the conversion by taking GPS coordinates and zone number as input. The software used in this work is a Python interface to PROJ.4 library [78] called pyproj [78]. Secondly, the ground truth of the SUAS positions were brought to the world frame by reference frame transformation.

$$\text{True SUAS position}^{W} = Q_{step2}^{-1} Q_{step1}^{-1} \cdot \text{SUAS position}^{UTM} \tag{4.25}$$

where Q_{step1}^{-1} and Q_{step2}^{-1} is the inverse transformation matrix for a reference frame evolving from UTM to world frame. The formulation for coordinate transformation is given in Appendix A. The UTM to world frame transformation is illustrated in Figure 4.4, and the transformation parameters for step 1 and step 2 are also listed.

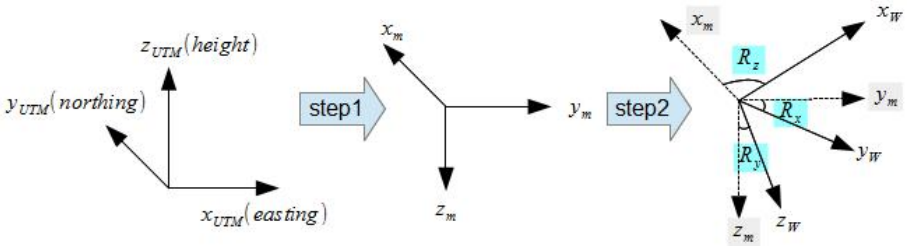

Figure 4.4: Coordinate transformation from UTM to world frame

$$T_{step1} = \left[Easting_{init}, Northing_{init}, Height_{init} \right] \tag{4.26}$$

$$R_{step1} = \left[180°, 0°, -90° \right] \tag{4.27}$$

$$T_{step2} = \left[0, 0, 0 \right] \tag{4.28}$$

$$R_{step2} = \left[Roll_{init}, Pitch_{init}, Heading_{init} \right] \tag{4.29}$$

where $Easting_{init}$, $Northing_{init}$, $Height_{init}$, $Roll_{init}$, $Pitch_{init}$, and $Heading_{init}$ are the SUAS's position and orientation in UTM at time 0.

Ground truth SUAS orientations were captured by the UAV navigation unit GS-111m. Both ground truth and estimated orientation were represented in aircraft principal axes. Hence, for the ground truth SUAS orientation to be compared to the estimated data, the only step is to subtract the orientation at time 0.

$$\text{SUAS orientation}_{\text{Ground truth}} = \text{Orientation}^{Nav} - \text{Orientation}_0^{Nav} \tag{4.30}$$

The estimated SUAS poses can be obtained from the world reference frame estimates in the filter state vector:

$$\text{Estimated SUAS poses} = [-O^C_{XYZ}, -W^C_{XYZ}] \tag{4.31}$$

To examine the accuracy of estimated landmark positions, DEM data were converted into the world frame following the same procedure as the SUAS position. Landmarks estimated from the CC-EKF-SLAM algorithm were converted to the world frame using the estimated SUAS localization results as reference frame evolving parameters. Let $[X_i^W, Y_i^W, Z_i^W]_k^T$ be the i^{th} landmarks coordinates in world frame at step k, then

$$\begin{bmatrix} X_i^W \\ Y_i^W \\ Z_i^W \end{bmatrix}_k = Q^{-1}(\boldsymbol{O}^{\boldsymbol{c}}_{\boldsymbol{XYZ,k}}, \boldsymbol{W}^{\boldsymbol{c}}_{\boldsymbol{XYZ,k}}) \left(\begin{bmatrix} x_i^C \\ y_i^C \\ z_i^C \end{bmatrix}_k + \frac{1}{\rho_{i,k}} \boldsymbol{m}(\varphi_{i,k}^C, \theta_{i,k}^C) \right) \tag{4.32}$$

Chapter 5

Results From Flight Data

This chapter discusses the results of applying the proposed CC-EKF-SLAM algorithm on realistic aerial video and navigation data. Firstly, a video of 400 frames was processed by the CC-EKF-SLAM algorithm. The convergence, consistency, and accuracy of the result were analyzed against ground truth, and described in Section 5.1-5.3. While analyzing pure natural scene video, it was difficult to judge the correctness of the correspondence between the estimated landmarks and the ground truth due to the lack of distinguishable visual features. Therefore, an another video where landmarks can be manually and distinctively matched was processed, and the result is summarized in Section 5.4.

The flight test result proved the feasibility of the CC-EKF-SLAM in mapping object at over 1000 meters distance. When equipped with a high resolution high dynamic range camera, the system would be able to produce high resolution model of the surrounding environment, and detect static obstacles such as tall trees, steep hills, electrical power line towers, etc.

5.1 Convergence Analysis

When a landmark was added to the filter, the landmark initialization point was initialized to the zero which is the origin of the camera reference frame. φ and θ were calculated directly from the landmark position on the image plane. Therefore they had relatively high accuracy, and should stay relatively constant. The only parameter that experienced a converging process was the inverse depth ρ, which was initialized to 0.01 for all landmarks.

5.1.1 Convergence of Inverse Depth

Figure 5.1 shows the $1/\rho$ plot for the entire 400 frames. Landmarks added at the 1^{st} frame are drawn in grey lines. Landmarks added at later frames are drawn in blue lines. Landmarks that drifted away are drawn in red line. The Landmark IDs are also shown on the right side of the plot for these drifting landmarks. The depth estimates go through rapid changes for several frames after initialization. Within approximately 20 frames, most inverse depth estimates have settled to stable values. Sharp spikes in the middle of the tracking are due to the addition of new landmarks into the filter. The estimated landmark distances ranged from 400 meters to about 1200 meters, confirming the algorithm's capability for estimating landmarks at great distance. On the other hand, some landmarks take a long time to settle, while some others never settle and drift away as tracking continues, such as landmarks 32, 13, and 53.

The non-converging landmarks are those located at the edge of a hill where objects with a big difference in distance meet in the image. As an example, the visual pattern of landmark 13 at frame 50, 100, 150, 200, 250, 300, 350 and 398 are shown in Figure 5.2. It shows that the initial visual pattern for landmark 13 is detected at

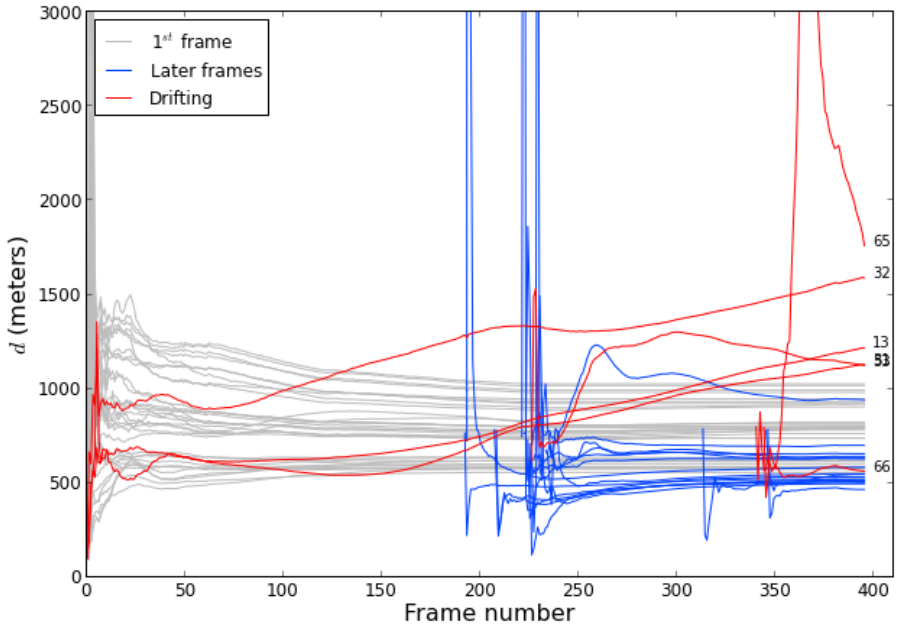

Figure 5.1: Inverse depth convergence

the immediate hill top. As the frame number advances, the camera view becomes different. The pattern comparison in the pyramid LK tracking algorithm normally allows for a small error at each iteration to tolerate the slight view change of the camera. This error accumulates slowly and eventually causes the tracked target to become a pattern located at a different location, which explains the slow drift in the depth estimate. To increase the reliability of the estimates, it is necessary to add a procedure into the algorithm that detect and handle such behavior.

5.1.2 Convergence of the Other Landmark Parameters

When landmarks were initialized, the initialization coordinates $[x_i, y_i, z_i]$, and the elevation-azimuth angle pair $[\varphi, \theta]$ do not go through a slow converging stage. Ideally, these parameters should stay at a relatively fixed value. As these parameters were used with the inverse depth to calculate landmark coordinates on the image plane

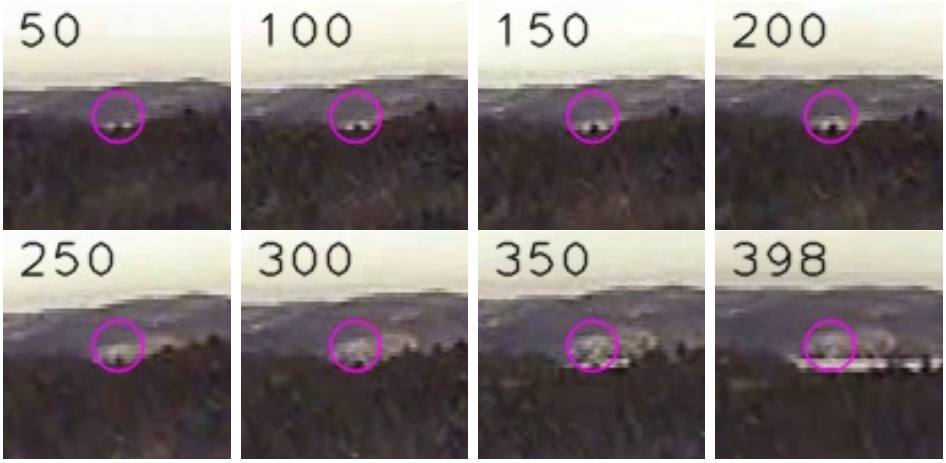

Figure 5.2: Landmark 13 at frame 50, 100, 150, 200, 250, 300, 350, and 398

and received a corrections at every iteration, their values were affected by the inverse depth, and varied at each iteration. The plots in Figure 5.3 show these parameters and their errors over the entire length of processed frames. The colors of the lines represents the landmark IDs, which were assigned sequentially when landmarks were initialized. The ground truth values of these parameters were extracted using the method described in Section 5.3.

The first column of the plots shows landmark initialization coordinates. The second column shows the errors of landmark initialization coordinates. All landmarks initialized at time 0 had stable values on initialization coordinates, with little variation and little error. Landmarks that were added at later frames showed drift from their initial values, as well as offset. The drift is a direct result of the filter correction process. When a new landmark was added to the filter, it had not established correlation with all the other landmarks and the world frame parameters, which caused its correction on the initialization point to be slightly different than all the existing landmarks and the world frame position. As a future improvement, it is reasonable for the newly added landmark to inherit the cross-correlation on the initialization coordinates from a landmark close to it. The biggest errors on landmark initialization

coordinates for landmarks added at later frames come from the offset errors. The offset errors are very significant on the Y axis. Since the landmark initialization coordinates are highly correlated to the SUAS location estimates, the offset errors are direct results of the errors in SUAS localization estimates. A more detail analysis is given in Section 6.2.2

The third and fourth columns of Figure 5.3 show elevation angle φ, azimuth angle θ, and their errors. For all landmarks, whether initialized at the first image frame or later frames, these angle estimates all diverged slightly from the initial values. There were two factors that contributed to the variation of φ and θ. Firstly, the initial depths of the landmarks were unknown. To compensate for the incorrect depth, the filter must adjust φ and θ estimates slightly so that the predicted measurements agree with the actual measurements. Secondly, big variations in the estimates were seen at around frame 200 when a lot of new landmarks were added to the filter. This observation suggests that an addition of new landmark has negative impact on the accuracy of estimates. Therefore additions of new landmarks should not be allowed to happen too frequently in order to limit their effects. For landmarks initialized at later frames, offset errors at initialization were much more significant, and were accounted for the majority of the mapping errors seen in those landmarks. Further discussion on the offset errors for these landmarks are presented in Section 6.2.2

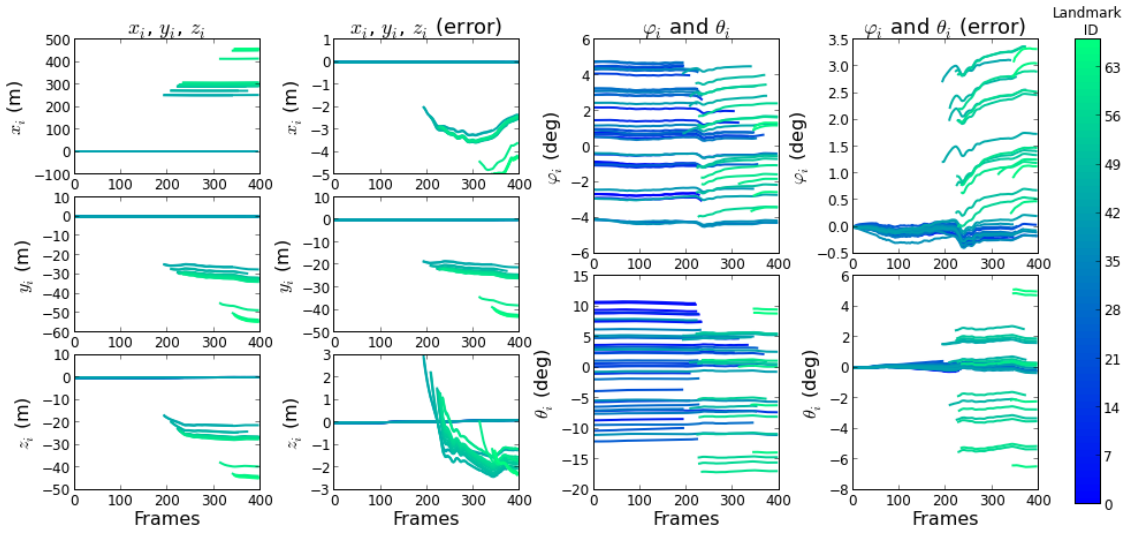

Figure 5.3: Convergence and errors of all other landmark parameters besides inverse depth

5.2 Consistency Analysis

Research on EKF SLAM properties indicated that

1. An EKF system becomes inconsistent when the variance of a state vector element becomes too small and forbids an effective correction,

2. Uncertainty of the world frame should increase as the camera frame moves away from the world frame origin.

To examine the consistency of the CC-EKF-SLAM algorithm, the variance of world frame parameters and camera motions are plotted in Figure 5.4, and the variance of landmark parameters are plotted in Figure 5.5. The variance of world frame position increases with iterations at the beginning. After frame 100, $\sigma^2_{O_Y}$ and $\sigma^2_{O_Z}$ converge to a relatively stable value. This is likely due to the small and slow variation on the Y and Z translation of the SUAS. On the other hand, camera rotation and world frame orientation variance have a similar pattern and amplitude. The X component of world frame orientation decreases with iterations, while Y and Z components

fluctuate around a fixed value and remain below 1.5e-5°. Landmark parameters φ and θ also have very small variance since they are affected by the variance of camera rotation. In the CC-EKF-SLAM algorithm, although the world frame orientation variance remains fixed at the prediction step, camera rotation variances have 0.1° of noise added (as indicated by the GS-111m specification [71]). An investigation into the algorithm shows that the EKF correction step reduces the noise of camera rotation significantly and results in a variance estimate much lower than the specification.

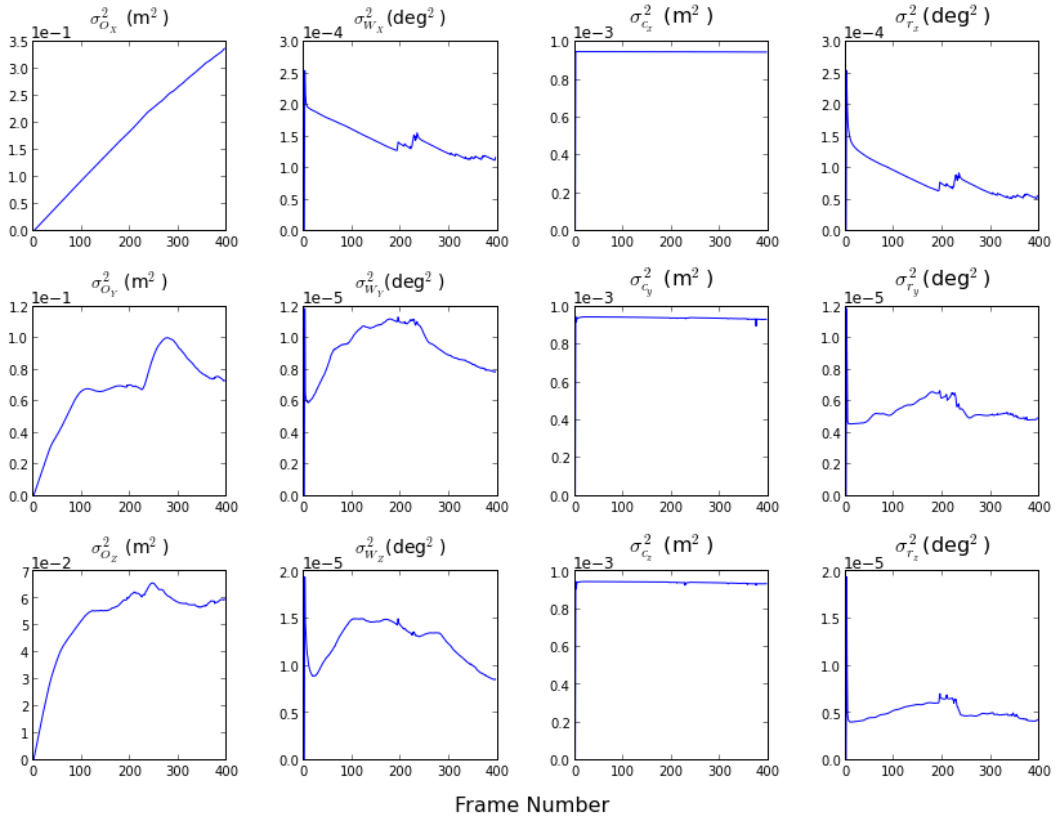

Figure 5.4: Variance of world frame parameters and camera motion

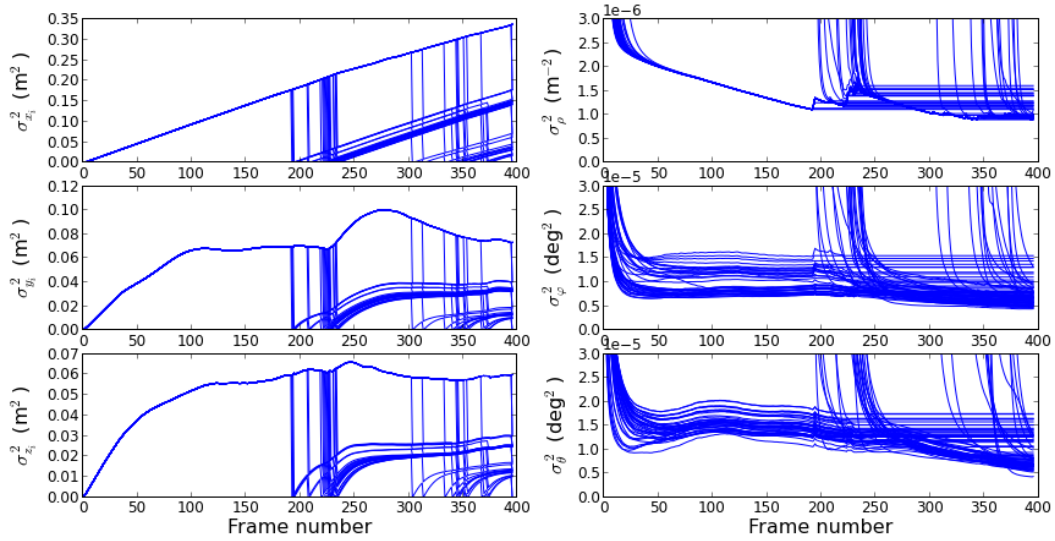

Figure 5.5: Variance of landmark parameters

To see the effect of small variance on the correction process of CC-EKF-SLAM, correction amounts applied to the predicted state vector are plotted in Figure 5.6. It can be observed that in general the correction amounts agree with the variance in that a bigger variance results in more correction, except for the world frame position and camera translation on Y and Z axes. These parameters receive more correction as tracking progresses despite their variance being stabilized or even decreasing. Secondly, there is a strong correlation between the landmark initialization positions, the world frame positions, and camera translation estimates. World frame positions and landmark initialization positions are inversely correlated to the camera translations on Y and Z axes. Thirdly, the camera parameter corrections show an obvious trend on Y and Z translation. The correction amounts on camera translation on Y and Z axes are increasing with time. Due to these relationships, the main contributor for drift in SUAS position and offset in landmark mapping should be the camera translation correction on Y and Z axes. On the other hand, the big correction on these parameters could be caused by the small variance in the camera rotation. If not enough

correction can be made into the camera rotation and φ and θ angles, the algorithm must compensate that by making more adjustment in the camera translation on the Y and Z axes. This issue requires further investigation, and will be analyzed in future work.

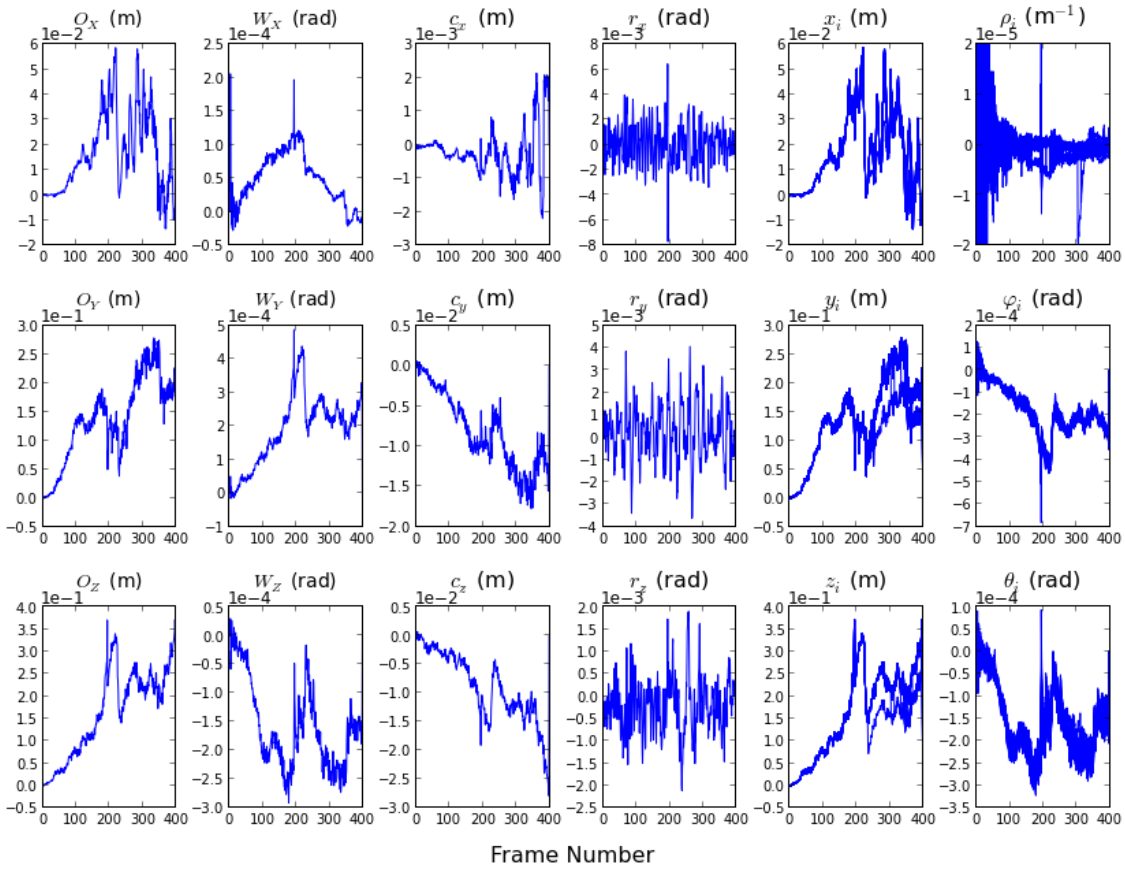

Figure 5.6: Plot of corrections applied to state vector

5.3 Accuracy Analysis

5.3.1 SUAS Localization

Ground truth for SUAS positions and orientations came from the GPS, and magnetometer. The GPS positioning can generally achieve 7.8 meters in accuracy [79]. For orientation, accuracy on the GS-111m datasheet specifies 0.1° for roll and pitch, and 0.5° for heading. Figure 5.7 shows the estimated SUAS position and orientation, the ground truth, and the error. From the comparison, X position error reaches a maximum of 4.8 meters, while error in Y and Z are bigger, reaching 60 meters and 28 meters respectively. The position on the Y axis is clearly diverging from the ground truth by the end of 400 frames. For orientation, the estimated value agreed with the ground truth pattern, with maximum error within 0.02 rad or 1.15°. There is no clear sign of divergence within the 400 frames processed. Another characteristic that can be observed is the correlation between the position errors and the aircraft orientation. Error on the Y position is correlated to the orientation on the Z axis; error on the Z position is correlated to the orientation change on Y axis. Clearly, aircraft rotational motion plays a crucial part on the accuracy of the estimates.

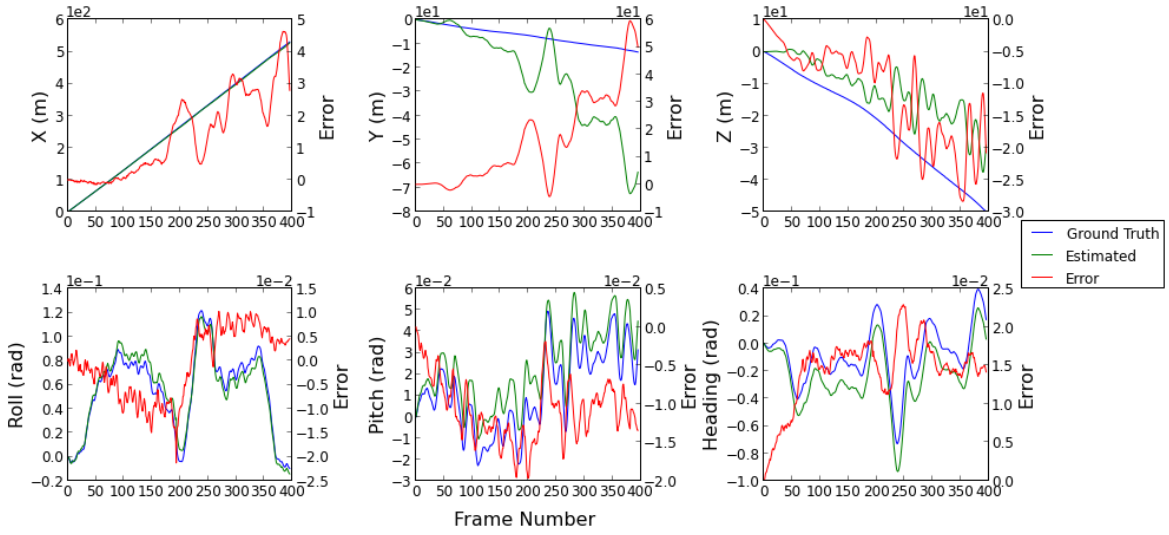

Figure 5.7: SUAS poses compared to ground truth for the natural scene video

5.3.2 Landmarks Mapping

The ground truth data came from the digital elevation map (DEM) which was downloaded from the CGIAR-CSI website [72]. The DEM has approximately 90 meters resolution, and the reported vertical error is less than 16 meters. The map was interpolated when being compared to the estimated landmark position.

Establishing a data correspondence between the DEM and the estimated landmarks is non-trivial for a natural scene which lacks distinguishable visual features such as building corners or other man-made objects. As shown in the converging plots Figure 5.3, landmark parameters do experience drift in flight. Hence, the best time for achieving data correspondence with the DEM is when landmarks are first initialized. At the time of initialization, landmark initialization positions $[x_i, y_i, z_i]$ were zeros. $[\varphi, \theta]$ were calculated from landmark coordinates in the image, and did not carry any error contributed by the unknown ρ or the pyramid LK algorithm. Therefore, the following procedure was used to find the corresponding ground truth location for every estimated landmark.

1. Identify the frame number at which a landmark is initialized. Record the ground truth position of the SUAS at that frame.

2. Create a reference frame aligned with the UTM frame on DEM using the ground truth SUAS position.

3. Convert the landmark parameters θ and φ angles at initialization from the camera frame to the reference frame created in step 2.

4. Create a vertical plane using θ to slice the DEM to form a 1D elevation plot. An example is shown by the blue line in Figure 5.8.

5. Create a line using φ to intersect the 1D elevation plot (Figure 5.8 green line).

6. The 1^{st} intersection is used as the ground truth landmark location.

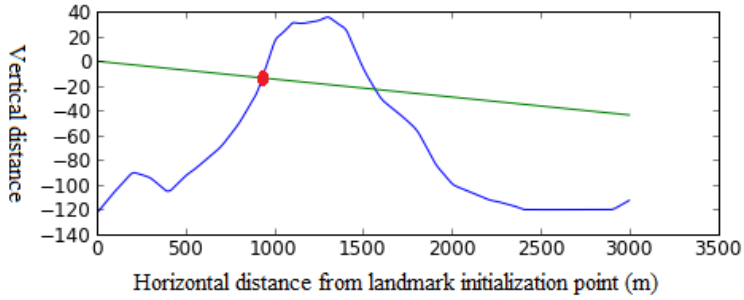

Figure 5.8: Finding ground truth landmark position on DEM

The convergence of landmark position errors is plotted in Figure 5.9. Landmarks added at the 1^{st} frame are drawn in grey lines. Landmarks added at later frames are drawn in blue lines. Landmarks that drifted away are drawn in red lines. The errors on the X component (along which axis the SUAS was traveling) converge to zero in general, with the amount ranging from 2.8 meters to 120 meters. The errors on the Y component show clear offsets for landmarks not initialized at the first frame. This

behavior is similar to what is seen in the error analysis in Section 6.2.2 where offset in landmark position estimates are caused by drift in UAV localization estimation. Offsets on the Z component are much less obvious.

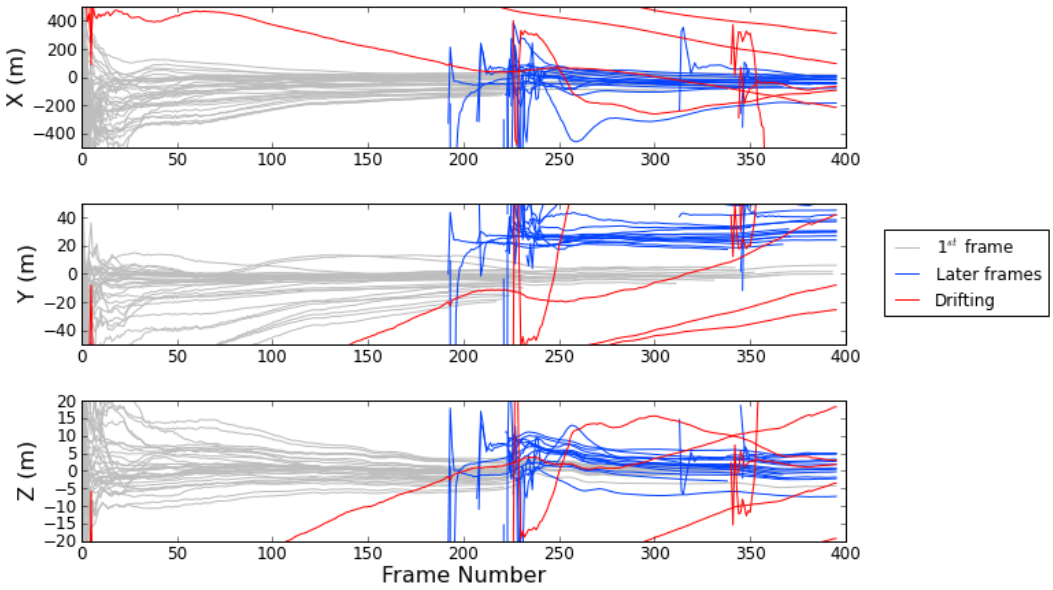

Figure 5.9: Landmark position errors convergence in world frame

Landmark IDs were assigned to landmarks sequentially when landmarks were initialized into the filter. Plotting the estimated and ground truth landmark positions against landmark IDs revealed more detail on the errors. Figure 5.10 shows the estimated and ground truth landmark positions on the left, and the errors on the right. On X and Z axes, the biggest error occurs on landmark 65 which was initialized fairly late in the sequence, and may not have converged properly. Other landmarks that carry big errors, such as landmarks 13, 31, and 32, are all located on the hill top in the image of Figure 5.11. The visual tracking problem discussed earlier applies to all these line features. Hence, accuracy of distance estimates on these landmarks were affected. On the Y axis, the plots show that landmarks with IDs higher than 40 have offset errors. All landmarks with IDs higher than 40 were initialized after frame

180, at which point the SUAS had travelled away from the world frame origin for at least 200 meters. In general, the average error increases as landmark ID gets bigger, which indicates that the further the landmark initialization point is from the world frame origin, the bigger the offset error becomes. A more detail analysis was done to find the source of these offset errors, and the results are presented in Chapter 6. The statistics of landmark position errors are summarized in Table 5.1.

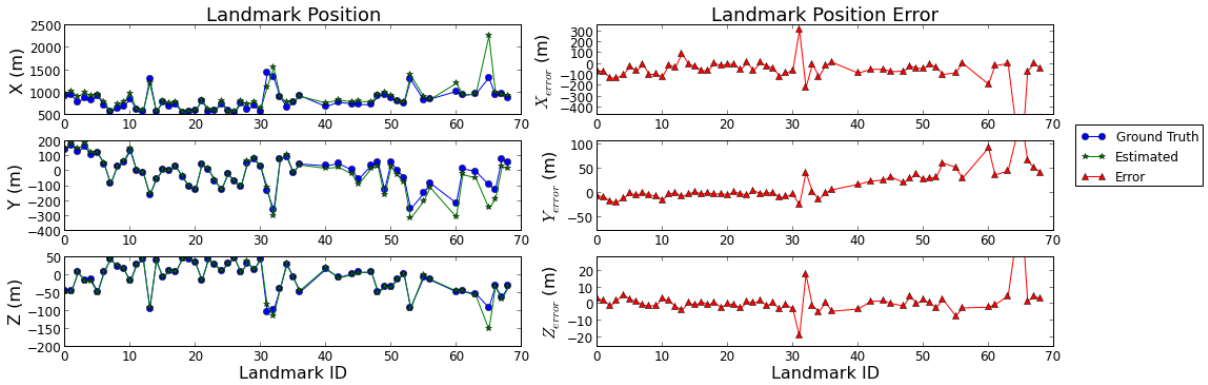

Figure 5.10: Landmark positions and errors plotted against initialization sequence for the natural scene video

Table 5.1: Landmark mapping error statistics for natural scene

	Mean	Standard Deviation
X_{error}	-56.21 meters	136.23 meters
Y_{error}	13.75 meters	31.01 meters
Z_{error}	1.34 meters	9.09 meters

Using the estimated landmark positions, a terrain map can be generated. Figure 5.12 shows the resulting terrain map on the left with landmarks marked by circle markers. On the right, the ground truth DEM is also shown for comparison. The estimated terrain map agrees with the DEM at location where sufficient amount of

Figure 5.11: Visual appearance of landmark 13, 31, 32, and 65

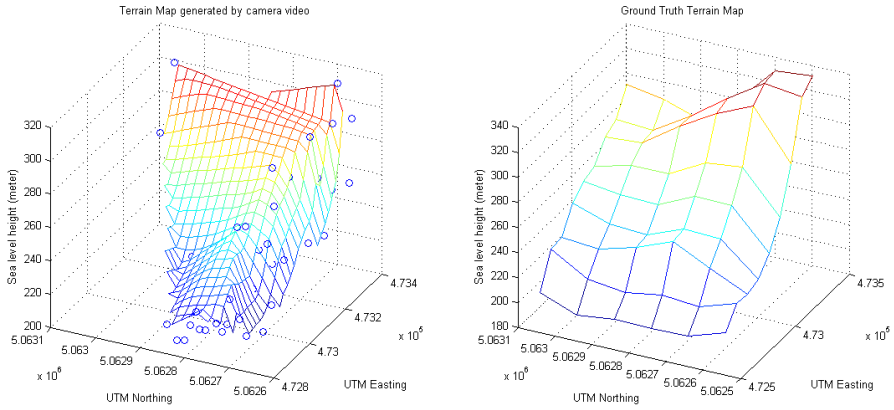

Figure 5.12: Terrain map comparison. Left: terrain map generated from estimated landmarks. Right: the digital elevation map (DEM).

landmarks can be established, such as the front and right side of the map.

5.4 Accuracy Verification through Manually Corresponded Landmarks

Due to the difficulty in corresponding estimated landmarks to the DEM, a second piece of video was processed with man-made structures in the scene so that landmark correspondence can be made manually. Ground truth on landmark coordinates were taken from Google Earth. Since Google Earth provides very rough elevation data, ground truth on landmark elevations were taken from the GPS measurements of the helicopter after it had landed at the airport. Figure 5.13 shows a zoomed-in view of

the landmarks extracted by the CC-EKF-SLAM. All landmarks are located at the bottom left corner of the image. Figure 5.14 shows the common landmarks found manually on the satellite image on Google Earth [80].

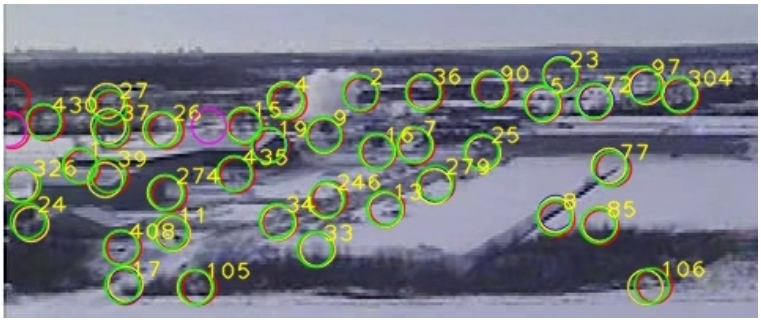

Figure 5.13: Landmarks extracted by algorithm from the airport landing video

Figure 5.14: Common landmarks manually identified on Google Earth

The estimated SUAS poses are compared to the GPS and IMU recordings in Figure 5.15. Similar to the natural scene video, SUAS position estimates are more accurate on X and Z, and experience more drift on the Y axis. The drift becomes significant when the frame number exceeds 200. The correlation between the position errors and the rotational motions of the aircraft can also be found in this plot.

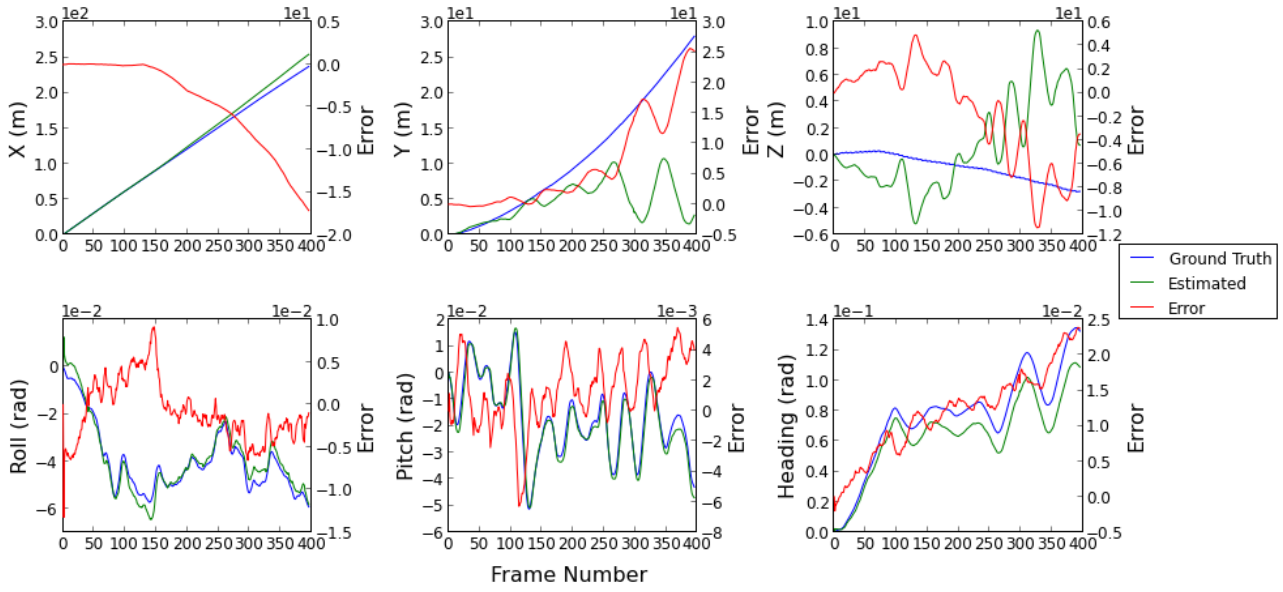

Figure 5.15: SUAS poses compared to ground truth for the airport landing video

Figure 5.16 shows the landmark positions compared to ground truth. This plot shows a clear offset error between the ground truth and the estimated. The error statistics are given in Table 5.2. Compared to the landmark error statistic from natural scene video, these errors have bigger mean, but much smaller standard deviation. This suggests that the landmarks might be affected by specific error source. Since the offset errors exist on all landmarks regardless of their ID value, and all landmarks are located at the bottom left corner of the FOV, it is likely that lens distortion is the main contributor for these errors. On the other hand, the error from airport landing video are within 3σ of the error statistics established in the natural scene video. The statistics comparison confirms that the accuracy analysis result from both videos agrees in general, and that the ground truth to estimated landmark correspondence method used in natural scene is valid.

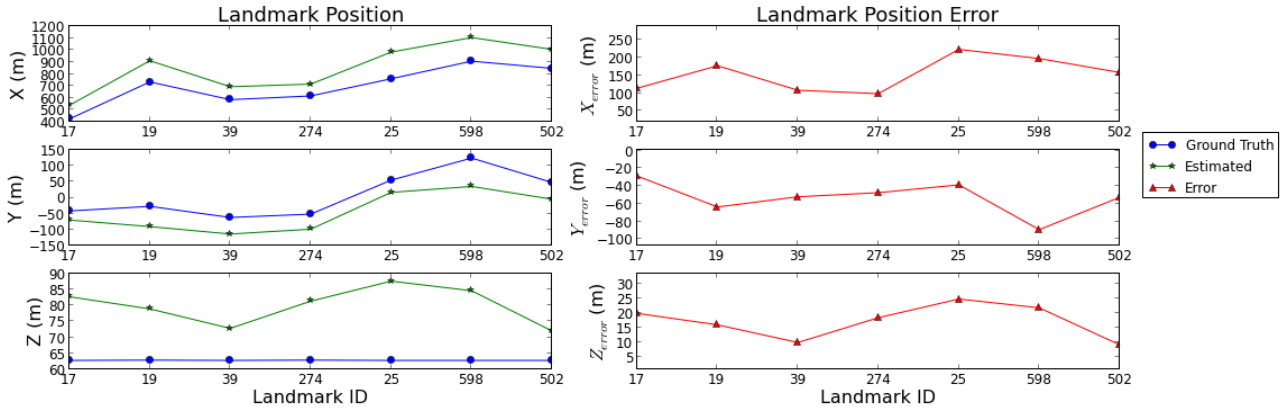

Figure 5.16: Landmark positions and errors for the airport landing video

Table 5.2: Landmark mapping error statistics for airport landing scene

	Mean	Standard Deviation
X_{error}	154.33 meters	44.86 meters
Y_{error}	-53.15 meters	17.98 meters
Z_{error}	17.27 meters	5.44 meters

Chapter 6

Error Analysis through Simulation

The flight test results proved the feasibility of using the CC-EKF-SLAM algorithm for distant object mapping. It also revealed there existed some error in the result of the algorithm, with error of up to 60 meters for SUAS localization and up to 150 meters for most landmarks positions. It is important to understand the cause of these errors so that improvement on the algorithm can be made.

There are many factors affecting the accuracy in UAV localization and landmarks mapping results, and they can be sorted into three main categories:

1. Error caused by the SLAM algorithm itself.

    - Firstly, the algorithm estimates landmark coordinates through a model that describes the relation between the UAV poses, the landmark positions in 3D world, and the landmark coordinates on the image plane. The model may not be describing every aspect of the relation fully. Any relation not included in the model introduces error into the result.
    - Secondly, as the model is non-linear, the linearization process can introduce error into the result. The amount of error may be different for different kind of maneuver. In the test flight, besides moving forward, the SUAS

do experience oscillatory rotation on all three axes.

2. Error in system intrinsic parameters estimates. The system intrinsic parameters include

   - Camera intrinsic parameters
     - Coordinate of optical center on the image plane $[c_x, c_y]$
     - Scaling factors that project landmarks in 3D world to the image plane $[f_x, f_y]$
     - Lens distortion parameters $[k_1, k_2, p_1, p_2]$
   - Image resolution
   - Accelerometer bias

3. Error introduced by pyramid Lucas-Kanade (LK) tracking algorithm. Firstly, LK tracking algorithm tracks landmarks by comparing the intensity of a windowed image centered at the landmark coordinate from one frame to another. The searching of the matching window terminates when the sum of squared error (SSE) on the windowed image is lower than a threshold set by the user, or when the number of iterations of the search has reached a maximum number (also set by user). As the scene evolves from frame to frame, the initial landmark appears differently as viewing distance and angle changes. Allowing a small SSE between matched windows is necessary to give tolerance for the matching. However, it could also cause the window to move in the neighborhood of the landmark coordinate, and eventually drift off, resulting in a loss of the original visual pattern. An example was shown in Figure 5.2 in Chapter 5. Secondly, sudden intensity changes in image sequences could result in significant noise in the tracking. In an outdoor setting, intensity change can be

introduced by many factors, such as changes of sky portion in an image, sun glare, UAV entering or exiting cloud shade, or the camera auto-adjusting its shutter speed, etc. In this work, error from illumination changes were reduced by feeding the pyramid LK algorithm with the predicted landmark coordinates on the image plane. No significant error can be observed due to illumination variation in the video processed. As a reliable vision tracking algorithm is an entire field of research in itself, it will not be discussed further in this chapter.

To better understand the impact from different UAV motion and error in system intrinsic parameters, a series of simulations were performed to examine factors discussed in item 1 and 2 independently. Result from the analysis will help improving the CC-EKF-SLAM itself, selecting appropriate hardware for the system, as well as quantifying the specification of the OD system.

The procedure described below simulates the entire process of perceiving landmarks through a digital camera under arbitrary UAV motion. The simulator first generates a 3D point cloud ranging between 100 meters to 1500 meters from the camera (Figure 6.22). At each frame, the coordinates of the 3D points are transformed to the new camera frame using a simulated UAV pose, which allows for studying the algorithm under various motions. Next, the 3D points are projected onto the image plane using a camera model defined by $[c_x, c_y, f_x, f_y, k_1, k_2, p_1, p_2]$, and digitized to a resolution of choice. The camera model used by the simulator may be different than the model used by the CC-EKF-SLAM algorithm, which allows for injection of camera calibration errors. The resulting coordinates of the 3D points were used directly as measurements in the correction step of the EKF. The simulator does not simulate errors contributed by the visual tracking algorithm, or by navigation measurements.

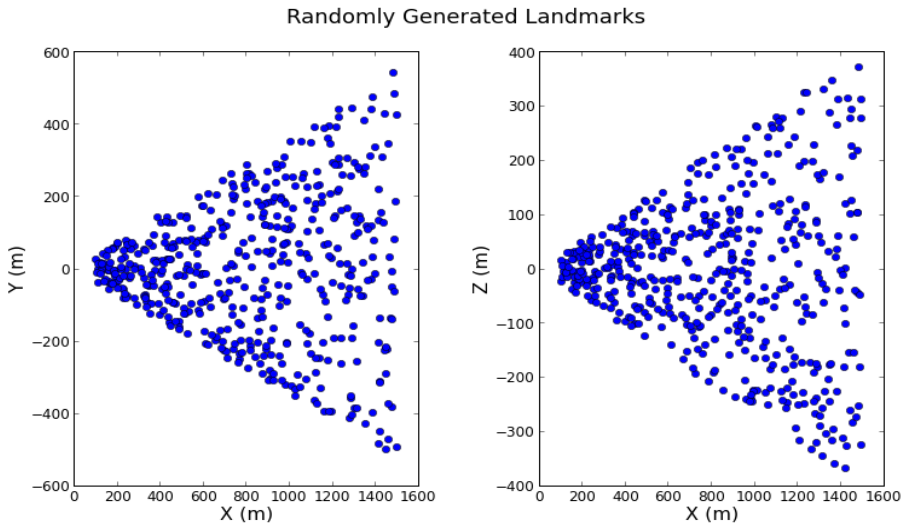

Figure 6.1: Randomly generated 3D landmark points

6.1 An Ideal Case

First of all, understanding the algorithm's performance under nearly no noise conditions provides a solid ground for the analysis later on. This simulation showed how much error the algorithm itself generated under the most basic flying condition, which is moving forward at constant speed. The low noise environment was configured as follows,

- UAV was moving forward (X axis) with constant speed of 60 knots.
- Y axis and Z axis translation were limited to white noise with standard deviation of 0.08 meters and a mean of 0.
- UAV rotations on all three axes were modelled by white noise with standard deviation of 0.01 degree and a mean of 0 degree.
- No image digitization error was introduced (i.e. the projected landmark position on the image plane was not digitized to any sensor resolution).

- No error was introduced from camera model mismatch (i.e. camera intrinsic parameters used by the simulator was exactly the same as those used in CC-EKF-SLAM algorithm).

6.1.1 UAV Localization

Accuracy of UAV poses estimates are plotted in Figure 6.2. The ground truth and estimated value are plotted in blue and green lines respectively. The error is plotted in red line. Under a simple forward motion, the algorithm tracks the UAV status quite well. Error on translational motion is less than 1 cm and error on rotational motion less than 3e-3 degree. An obvious behavior of the error is that the error starts from 0, experiences some rapid change during the first few frames, and settles to a relatively stable value. This matches the converging behavior of the landmark inverse depth ρ which was shown next. This observation suggests that although the inverse depth representation is more linear than th direct depth representation, it has a negative impact on the accuracy of the estimates when being included in the filter vector before its approximate value is known. Hence, delaying integration of inverse depth till its value is approximately known should increase the accuracy of the state vector estimates. As triangulating a landmark requires 2 frames in a monocular vision setup, the delay requires only one extra frame, and is insignificant in most scenarios.

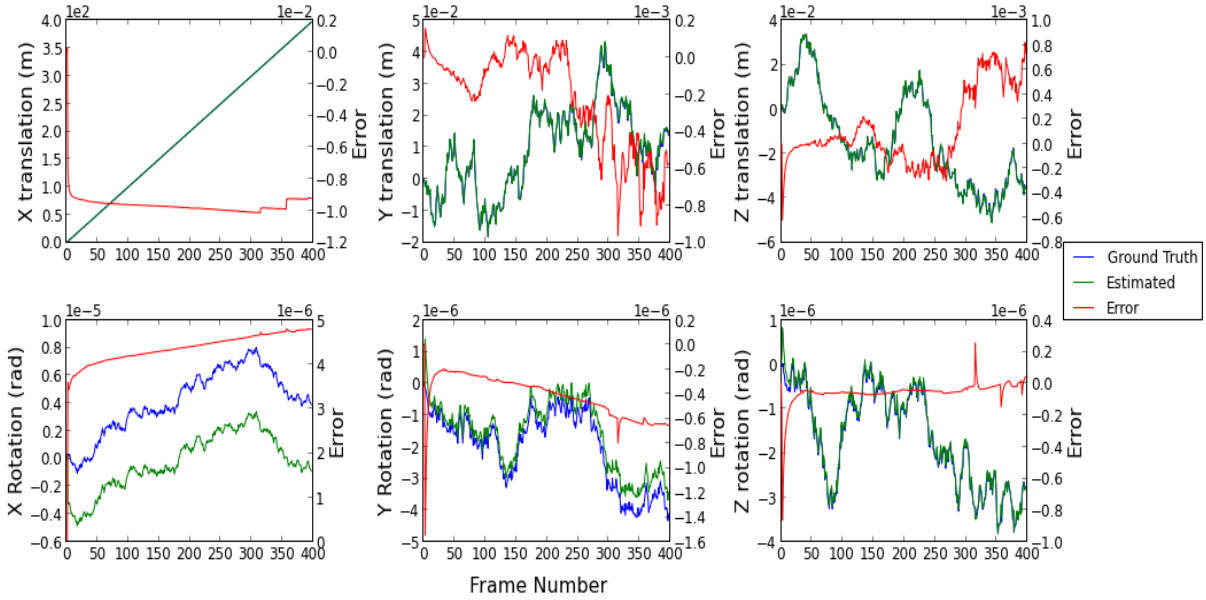

Figure 6.2: UAV localization errors under low noise condition and forward motion

6.1.2 Landmarks Mapping

Figure 6.3 top left shows landmark parameters $[d, \varphi, \theta]$ (where $d = 1/\rho$) plotted against frame number for the first 50 frames. The landmark depth d for all landmarks converges within the first 3 frames. The elevation-azimuth angles $[\varphi, \theta]$ stay almost constant after initialization. More detail can be seen from the error convergence plot for these parameters, plotted in Figure 6.3 top right. The error of landmark distance d slowly converges to zero. $[\varphi, \theta]$ errors starts from 0 at frame 0, and converge to a small offset while landmark depth d converge. As discussed in Chapter 5, in order to compensate for the inaccurate estimates of depth at filter initialization, the EKF makes adjustments to φ and θ so that the resulting coordinates of landmarks agree with the measurements. Therefore early integration of landmark depth also has negative impact on the accuracy of landmarks positions estimation.

The errors for landmark coordinates in world frame are plotted against landmarks IDs and frame number in Figure 6.3 bottom left and right. The landmark coordinate

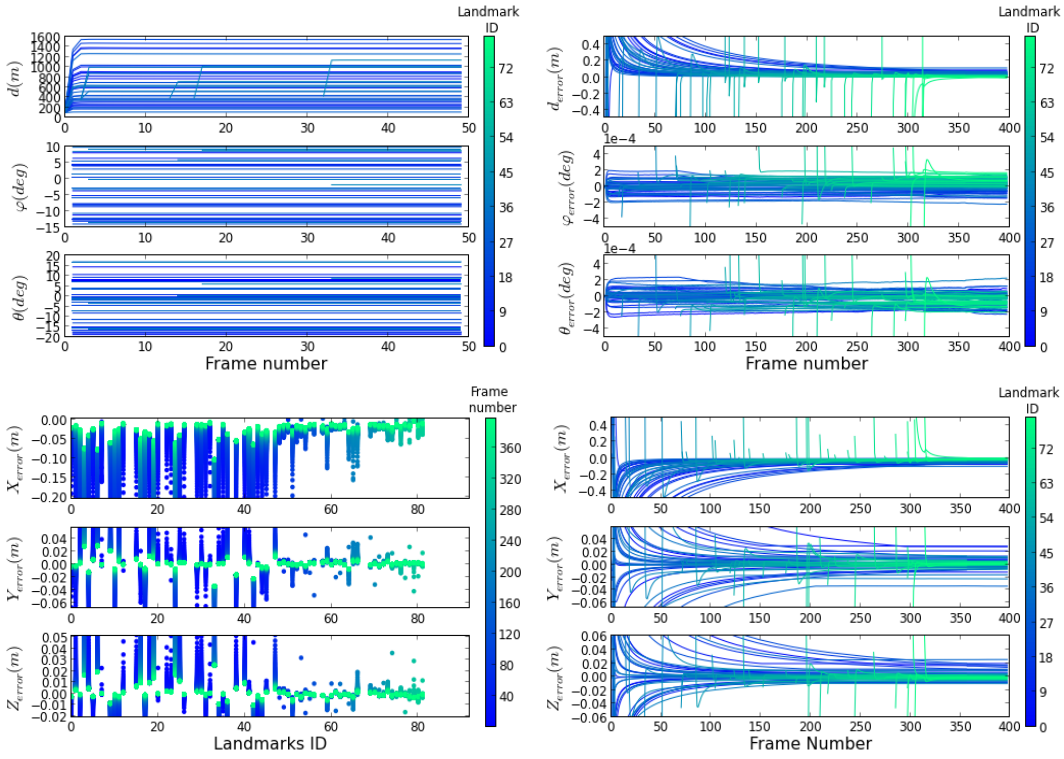

Figure 6.3: Landmark parameters and errors convergence under low noise condition and forward motion

errors in world frame all converge toward zero. During the 400 frames period, some landmarks move out of the FOV, and therefore their position estimates remain unchanged afterward. At the end of the 400 frames, landmark coordinate errors on X axis are reduced to +/-0.2 meters; on Y and Z axes, errors are reduced to +/-0.02 meters.

The findings of simple forward motion analysis are summarized below.

- Early landmark integration causes offset errors in UAV self-localization, and $[\varphi, \theta]$ estimates due to the EKF compensating for landmark depth at integration.

- Error caused by CC-EKF-SLAM non-linearity under simple forward motion is very small, and negligible compared to the object distance. This observation indicates that when analyzing other factors, such as oscillatory motion or camera

calibration error, any significant error that appears would be due to the added factor, and not the forward motion.

6.2 Effect of UAV Oscillatory Motion

The simulation of UAV forward travel shows that the CC-EKF-SLAM algorithm does landmark tracking and self localization quite well under simple motion. Next, the algorithm was tested with a more complex and realistic scenario. The remaining 5 types of maneuvers were examined independently. These maneuvers are: translation on Y, translation on Z, rotation on X, rotation on Y and rotation on Z. In the test flight, motions on these 5 DOFs were mostly oscillatory, especially the rotations. Each motion was modelled by a Sine wave with frequency at 1Hz with variable amplitude.

$$\text{Added motion} = A_{\text{motion type}} \cdot sin(2\pi x + \pi/2)$$

Motion type is denoted by v_y, v_z for translation on Y and Z axes; w_x, w_y and w_z for rotation on X, Y, and Z axes. For translation, the Sine amplitude A_{v_y,v_z} varied from 1 meter to 19 meters with 2 meter increments. For rotation, the amplitude A_{w_x,w_y,w_z} varied from 0.001 radians to 0.018 radians (0.057° to 1.031°) with 0.001 radian increments. By setting different amplitude of the Sine function, a wide range of rate of change for specific motion can be simulated.

6.2.1 UAV Localization

Figure 6.4 shows the UAV localization error statistic under oscillatory translation on Y and Z axes and oscillatory rotation on X, Y, and Z axes. The blue dots marks the mean value μ of the error averaged throughout 400 frames of tracking, and the error bars mark the standard deviation σ.

The translation motion clearly increases the error of UAV localization. However, the amount is insignificant. With the Sine amplitude increased to 19m, the UAV position error increases by less than 0.02 meter.

On the other hand, rotational motions have a big impact on the accuracy of localization. Rotations around the X axis (roll) have small effect. No obvious increase in mean and standard deviation can be observed. Rotations around Y and Z axes yield significant errors.

- Rotation around the Y axis increases the mean and standard deviation of UAV position error on X. For UAV position on Z, the mean error stays zero, but standard deviation increases dramatically.
- The same behavior can be observed on the X and Y axes position error for rotation around the Z axis.

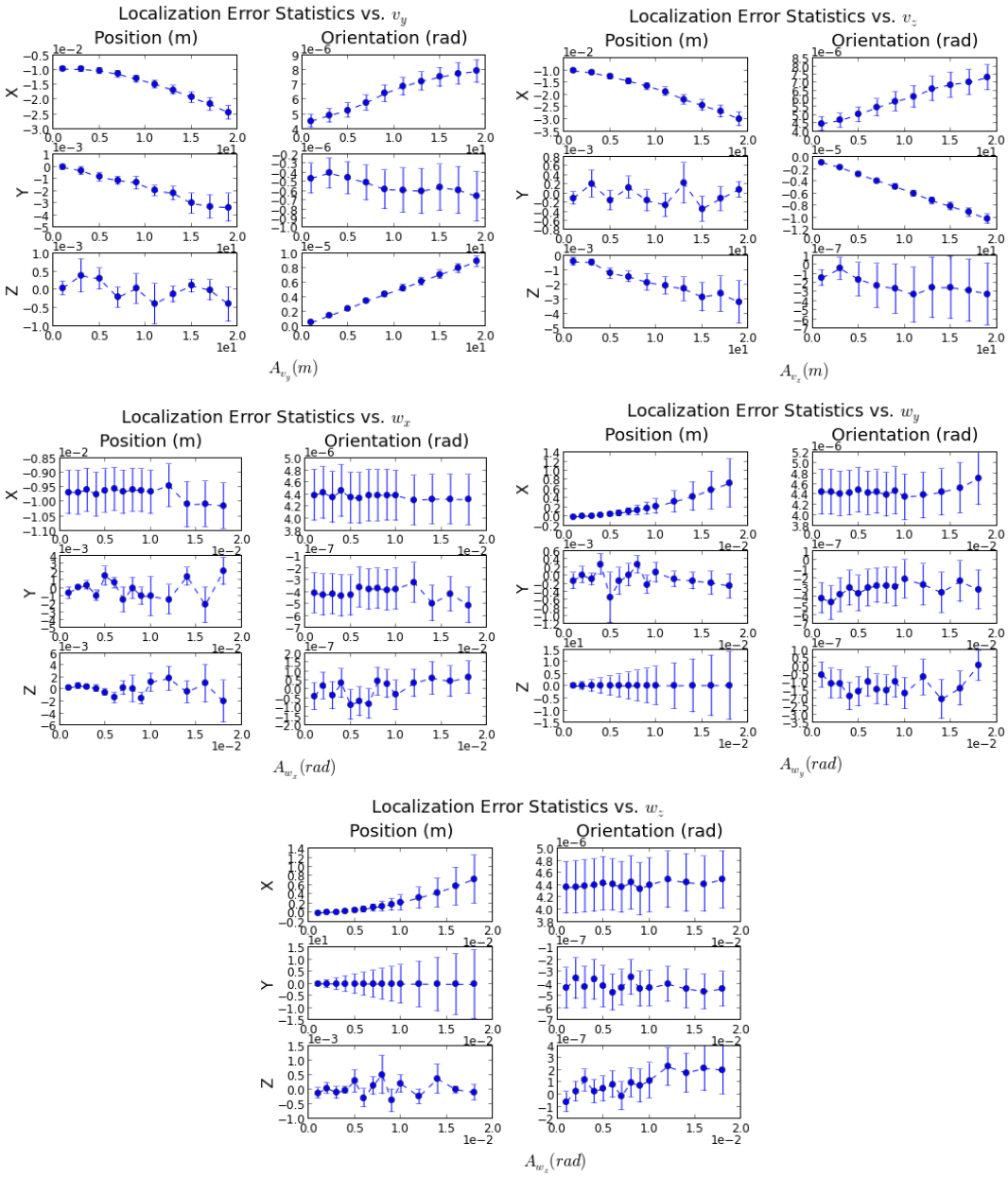

Figure 6.4: UAV localization error statistics under oscillatory motion

To understand how rotation affects the UAV localization estimates, UAV position errors in world frame are plotted below (Figure 6.5) with A_{w_x,w_y,w_z} set to 0.01 radians. When rotation occurs around the X axis, the position error of the UAV shows some oscillation on Y and Z axes. The oscillation magnitude remains small (on the scale of millimeters) and around zero. Hence, it does not generate significant mean and standard deviation change on the error statistic plot. For rotation around Y and Z, increase of rotational motion causes an increasing oscillatory error on the UAV position estimates (diverging). The mean UAV position error on the X axis increases in positive direction with rotation around both Y and Z axes. The most significant impact happens on the Z axis position (for rotation around Y) and Y axis position (for rotation around Z), with errors reaching 20 meters peak-to-peak at the end of the 400 frames. With an increasing rate of rotation, the rate of error diverging from zero also increases, and results in an increasing error standard deviation in the error statistic plots. This result suggests that the CC-EKF-SLAM algorithm is very sensitive to rotational motion around Y and Z axes.

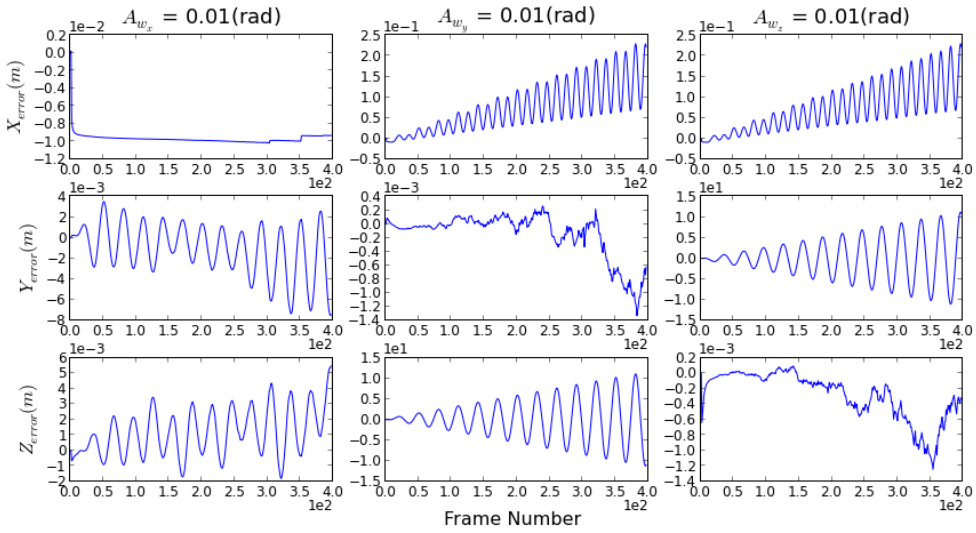

Figure 6.5: Estimated UAV position in world frame with oscillatory rotation, $A_{w_x,w_y,w_z} = 0.01rad$

6.2.2 Landmarks Mapping

Landmark mapping error statistics were calculated from the landmark errors at the last frame. Figure 6.6 shows the landmark mapping error statistics with added motions.

Translational motions increase both the mean and standard deviation of landmark mapping errors, but not by much. With A_{v_x,v_y} ranging from 1 meter to 19 meters, the increases in mean and standard deviation are both on the scale of centimeters.

Rotational motion has much bigger effect on the accuracy. With A_{w_x,w_y,w_z} ranging from 0.001 radians to 0.018 radians (0.057° to 1.031°)

- Rotation around the X axis causes a small standard deviation increase in the errors of landmark positions on the Y and Z axes. The errors are on the scale of meters under the maximum rotation setting.

- Rotation around the Y axis causes mean and standard deviation increase in landmark position errors on the X and Z axes. Landmark positions on Z receive the biggest impact with errors on the scale of hundreds of meters under the maximum rotation setting.

- Rotation around the Z axis affects landmark mapping on X and Y axes in a similar way as the Y axis rotation does.

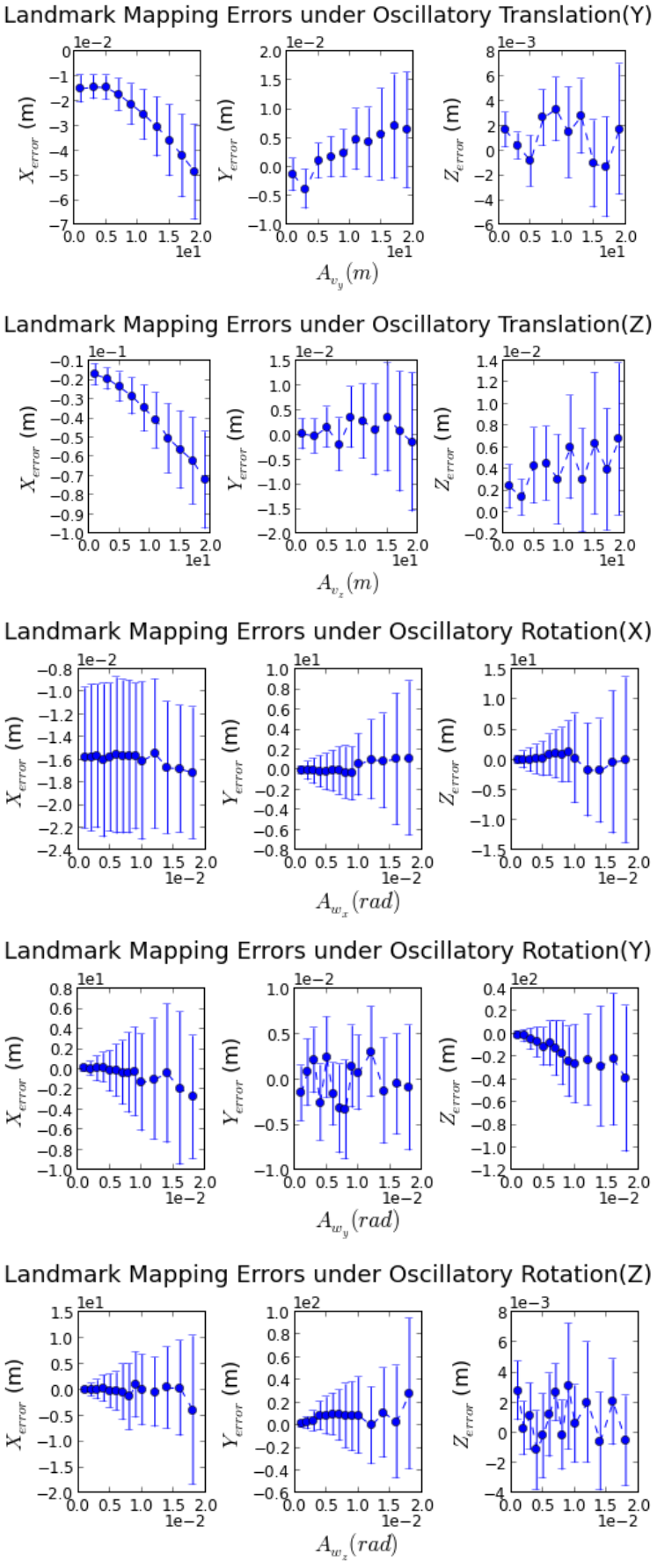

Figure 6.6: Landmark mapping error statistics under oscillatory motion

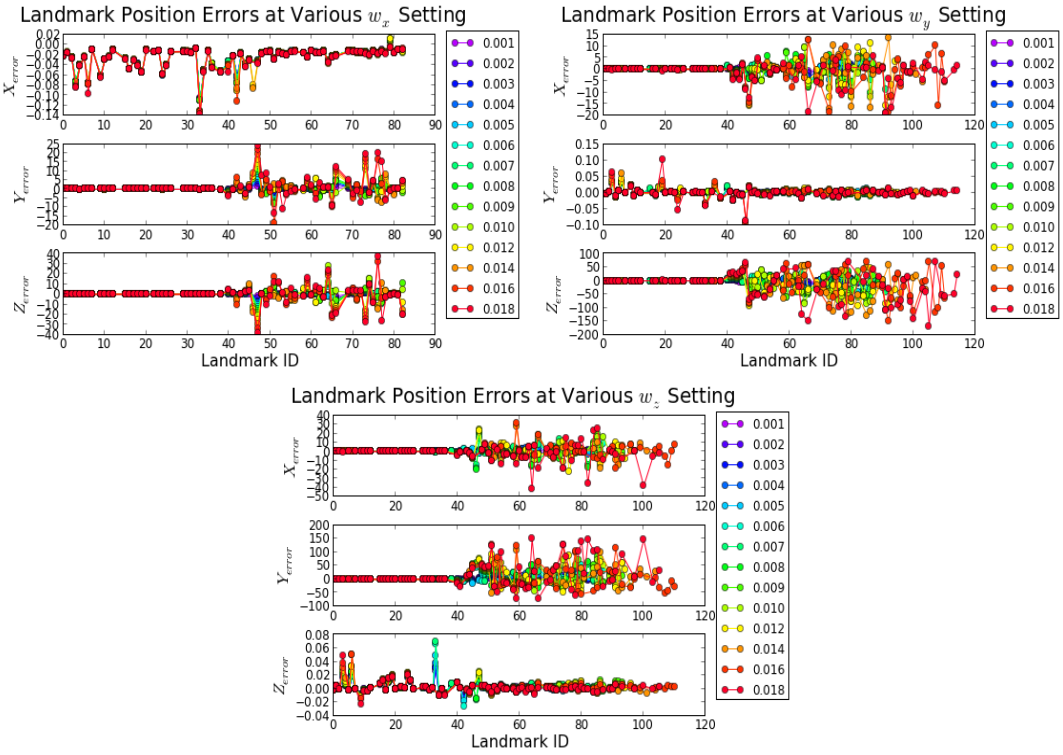

Figure 6.7: Landmark mapping errors plotted against initialization sequence under oscillatory rotation

Figure 6.7 shows landmark position errors at the last frame plotted against landmark IDs (initialization sequence), with line and marker color indicating rotation amplitude setting. The following observations can be made from the plots:

- With rotation around Y and Z, tracked landmarks were easily lost as they move out of the FOV. This can be observed from the increase of landmark IDs when Y axis rotation setting is increased from 0.001rad to 0.018rad. A frequent addition of new landmarks has a negative impact on the overall result due to the correlation between landmarks.

- Landmarks added after the first frame have a much bigger error than landmarks added at the first frame. All three plots in Figure 6.7 show that major landmark

mapping errors come from landmarks added after the first frame with ID bigger than 40.

To investigate how oscillatory rotations resulted in bigger error for landmarks added after the first frame, the scenario of UAV experiencing rotation around the Y axis with $A_{w_y} = 0.01rad$ was analyzed. Landmark parameters were converted into the world frame and their errors are shown in Figure 6.8. It was found that the most significant error occurs on parameter φ which is the landmark elevation angle. This angle has the same definition as the rotation angle around the Y axis. The second biggest error contributor is z_i, the landmark initialization coordinate on Z axis. Both parameters φ and z_i had an offset error at initialization, and were never corrected by the filter throughout the tracking.

Regarding the offset error in z_i, as discussed in Section 6.2.1, UAV localization estimates have the biggest position error on the X and Z axes under Y axis rotation. All landmarks not initialized at the first frame are affected by the UAV localization error, since the initialization point and UAV position are correlated.

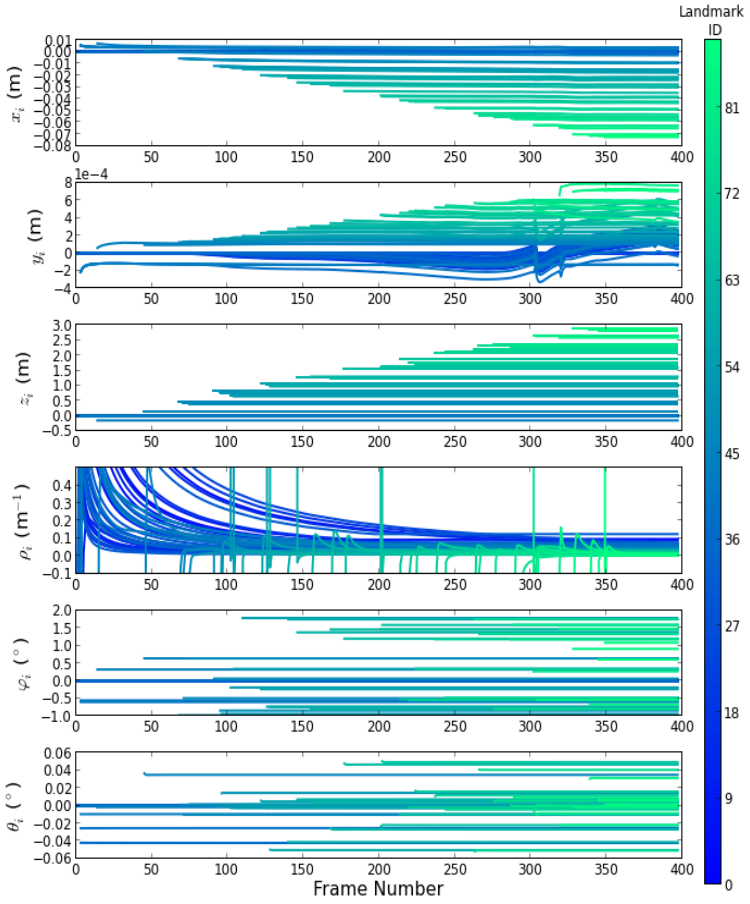

Figure 6.8: Landmark parameter errors under oscillatory rotation around the Y-axis, $A_{w_y} = 0.01rad$

Figure 6.9 shows the φ error at initialization in the camera frame and the world frame. The blue line shows the error in camera frame. The red line shows the error in world frame. As φ error is nearly 0 in the camera frame, it is clear that the reference frame transformation from camera frame to world frame introduced the offset error. During reference frame transformation, ground truth landmarks are transformed using the ground truth UAV poses; estimated landmarks are transformed using estimated UAV poses. Hence, even though the parameters are the same in the camera frame, results in the world frame are different. Landmarks initialized at the first frame do not carry any offset error because the reference frame transformation uses the same

parameters in both ways. During tracking, these landmarks are transformed to the new camera frame using the estimated UAV position and orientation. These are the same values being used to transform landmark positions from the camera frame back into the world frame. Therefore, although the UAV localization estimates are different from the ground truth, landmarks initialized at the first frame are not affected.

To conclude, the main contributor for landmark mapping errors come from errors in UAV localization estimation.

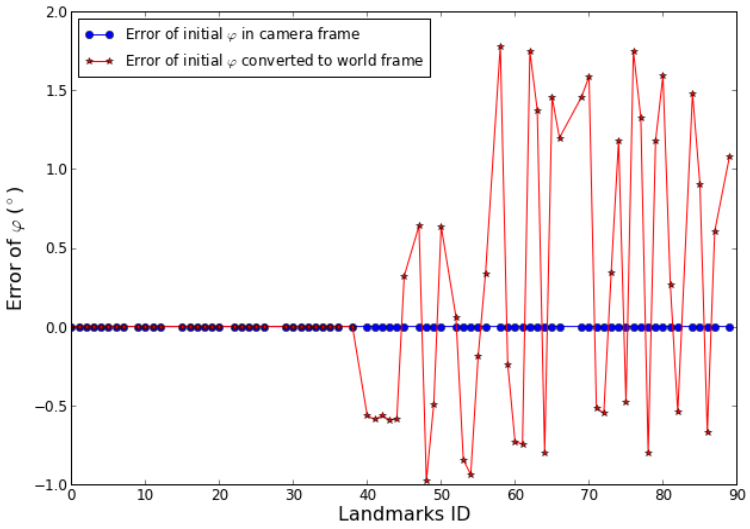

Figure 6.9: Error of φ at initialization in camera frame and world frame

6.3 Effect of Errors in Camera Intrinsic Parameters

This section discusses the effect on CC-EKF-SLAM accuracy from calibration errors of camera intrinsic parameters. Errors in camera intrinsic parameters were simulated by using slightly varied camera models in the simulator. c_x, c_y, f_x, and f_y were simulated individually and distortion parameters $[k1, k2, p1, p2]$ were simulated as a

group. Using the calibration result (see Section 3.2) as a base model, c_x, c_y, f_x, and f_y in the simulator varied from -50% to 50% of the base model. Lens distortion parameters varied from 0% to 120% of the base model.

6.3.1 Effect of Errors in (c_x, c_y)

Figure 6.10 shows an overview of UAV pose error statistics with calibration error introduced into (c_x, c_y). From the statistic plot, it was found that c_x had the biggest impact on UAV position on X and Y, and c_y had the biggest impact on UAV position on X and Z.

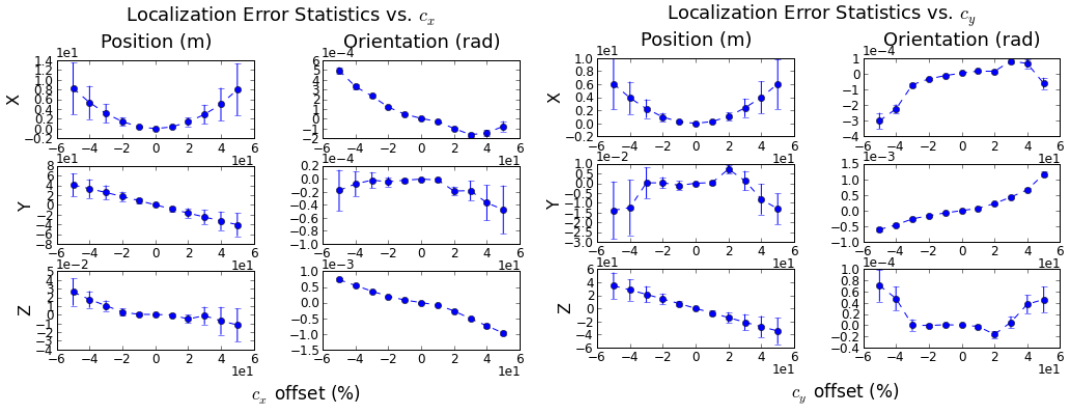

Figure 6.10: UAV localization error statistics with various (c_x, c_y) settings

Figure 6.11 shows the UAV position errors with c_x and c_y at various offsets from the base value. The resulting UAV position errors are dependent on (c_x, c_y) and are divergent in time. The divergence can be modeled by a first order polynomial function, with the rate of divergence decided by the offset of (c_x, c_y) from the base value.

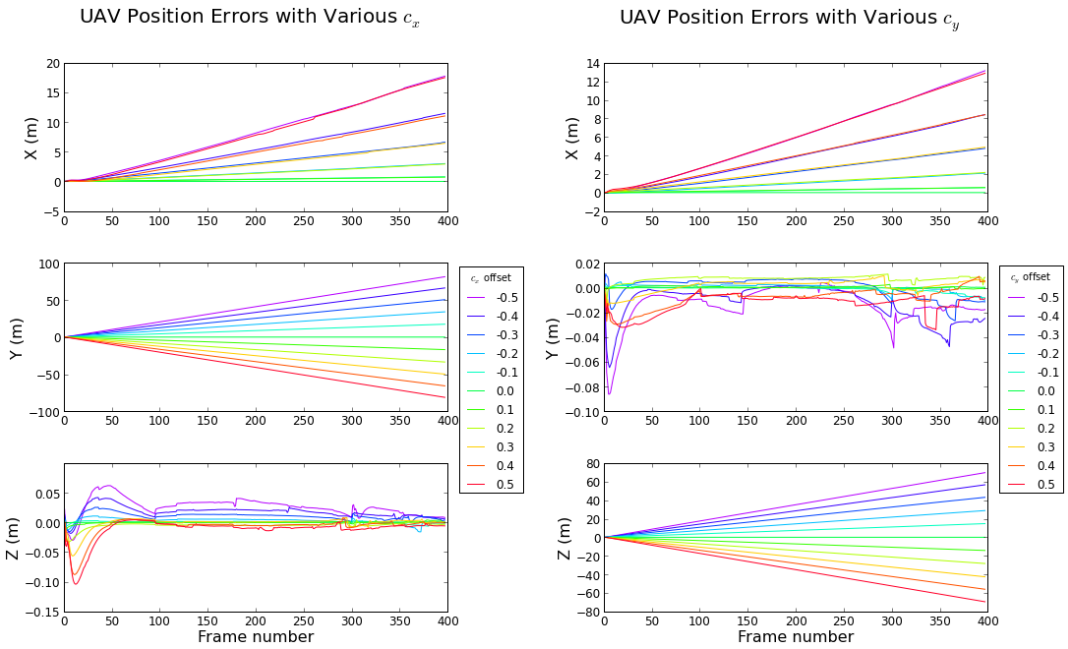

Figure 6.11: Diverging UAV position errors with error in(c_x, c_y)

The statistics of the landmark mapping errors due to the calibration errors in (c_x, c_y) are plotted in Figure 6.12. The following characteristics can be observed from the plots:

- Calibration error in c_x affects landmark positions on all axes, among which, the X and Y axes show the most significant error.
  - The further c_x deviates from the base value, the further the landmarks appear on average (positive mean error on X).
  - Mean landmark position errors on the Y axis are proportional to the calibration error in c_x. Their relation can be modeled by a first order polynomial equation.
  - Error in c_x also affects landmark positions on the Z axis, but on the scale of a few meters.

- Calibration error in c_y affects landmark positions on all axes similarly to c_x. Estimates on the X and Z axes show the greatest amount of error.

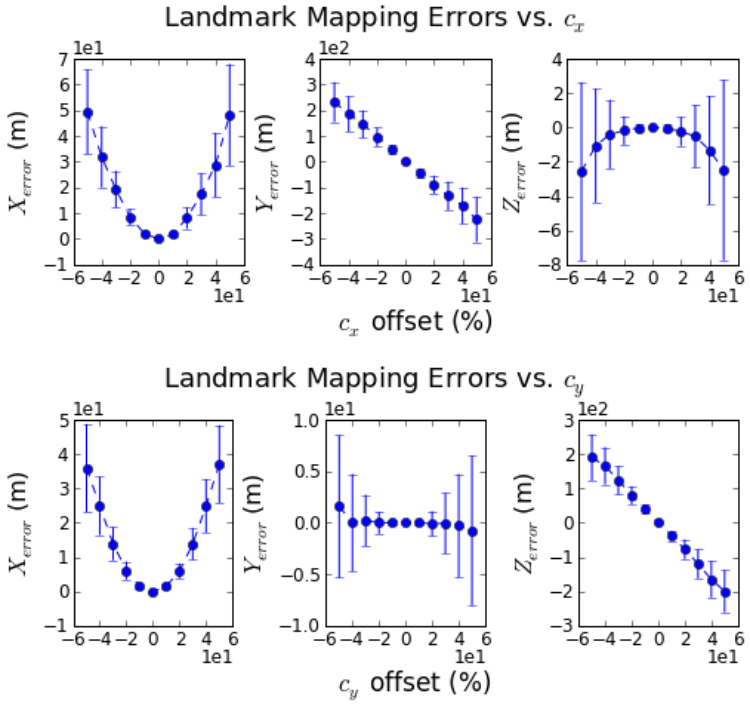

Figure 6.12: Landmark mapping error statistics with various (c_x, c_y) settings

Plotting landmark position errors against landmark ground truth positions reveals more information on how c_x and c_y affects landmark mapping. Landmark position errors are dependent on their ground truth positions, and the amount of error in c_x (or c_y).

- Landmark position errors on the X axis are proportional to the landmarks ground truth positions on X. The further the landmark is, the greater the error. The amount of error in c_x determines the slope of the error plot. The greater the error is in c_x, the steeper the slope becomes (Figure 6.13, top plot, subplot $[1, 1]$).

- Landmark position errors on the Y axis are also proportional to their ground truth positions on X with the slope and polarity dependent on the amount and polarity of the error in c_x (Figure 6.13, top plot, subplot $[2, 1]$).

- Landmark position errors on the Z axis are proportional to their ground truth positions on Z, with slope and polarity dependent on the amount and polarity of error in c_x (Figure 6.13, top plot, subplot $[3, 3]$).

- Error in c_y affects landmark positions similarly to c_x, but on different axis (Figure 6.13, bottom plot).

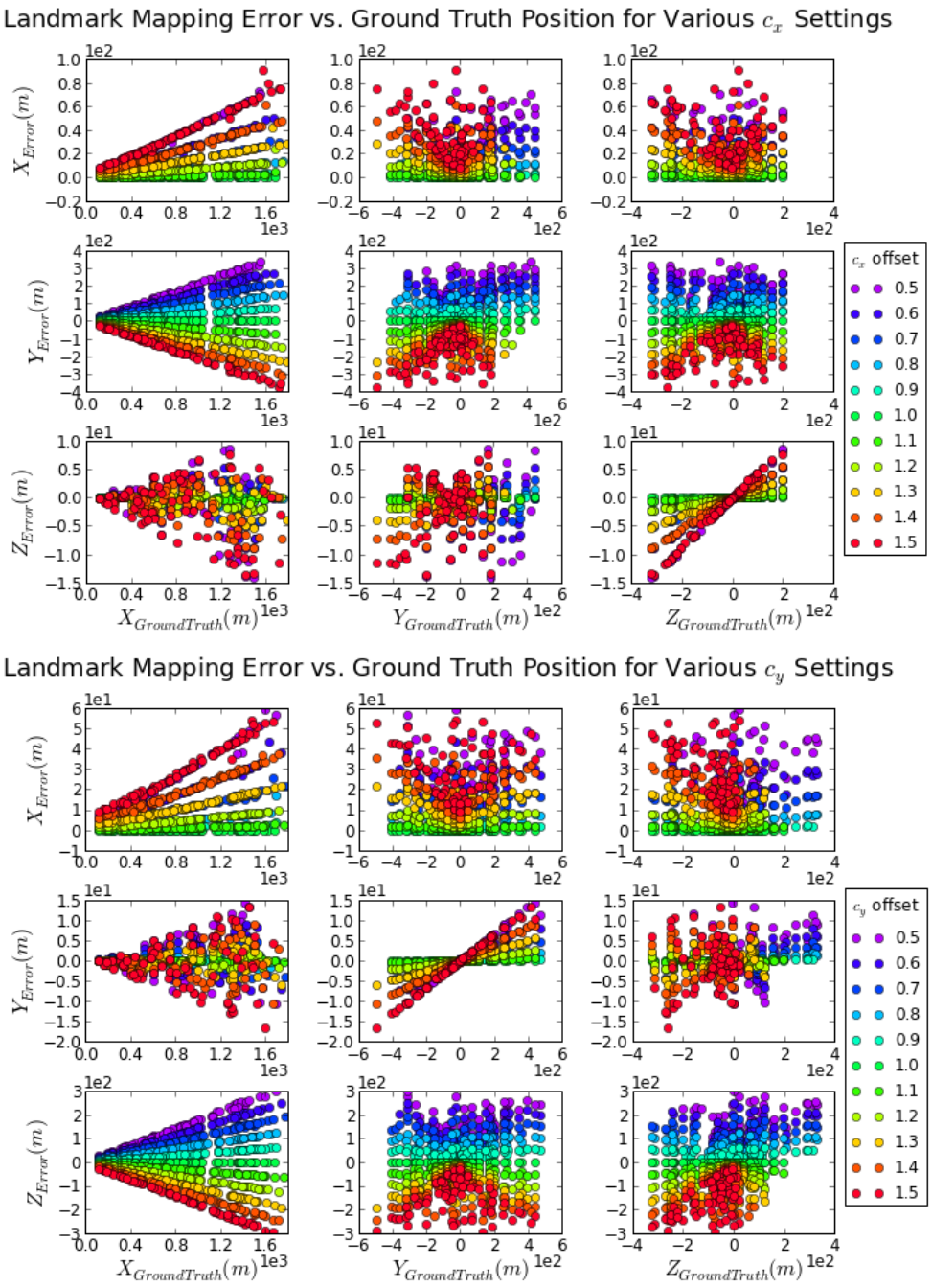

Figure 6.13: Landmark mapping errors versus ground truth landmark positions for various (c_x, c_y) settings

6.3.2 Effect of Errors in (f_x, f_y)

With (f_x, f_y) varying from -50% to +50% of the base value, the UAV localization errors are shown in Figure 6.14. For all f_x and f_y settings, UAV position errors remain less than +/-0.05 meters, and orientation errors remains less than 8e-6 radians. Compared to the errors obtained from the ideal case, calibration errors in (f_x, f_y) introduce minor errors into UAV localization estimates.

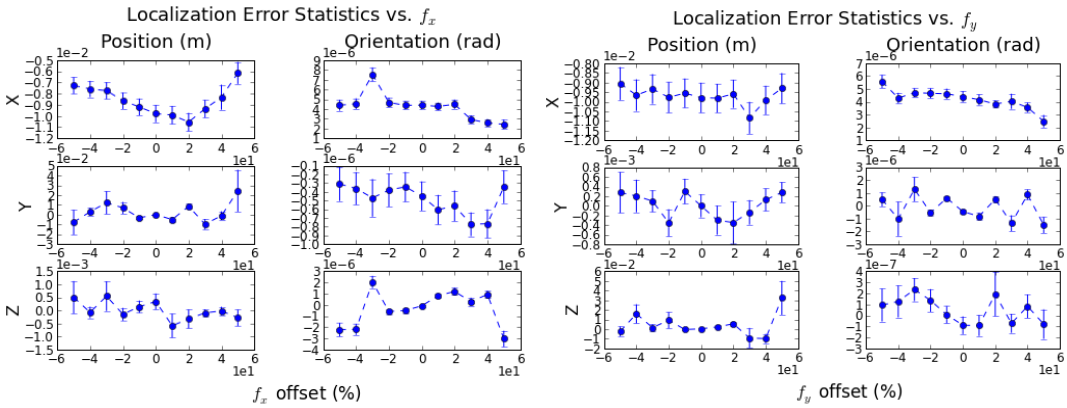

Figure 6.14: UAV localization error statistics with various (f_x, f_y) settings

On the other hand, landmark mapping errors are unavoidably affected by the calibration errors in (f_x, f_y) since these are the scaling factors that project landmarks from the 3D world onto the image plane. Figure 6.15 shows the error statistics of landmark position estimates under various (f_x, f_y) settings. The error on the X axis is minimal and is on the scale of millimeters. Errors on the Y axis receive the most impact from error in f_x, since this is the scaling factor that maps the Y component of landmark position in the world frame onto the u-axis on the image plane by $u = Y/X \cdot f_x$. The same behavior happens on the landmark position estimates on the Z axis and its relation to f_y.

Plotting the landmark mapping errors against landmark ground truth positions

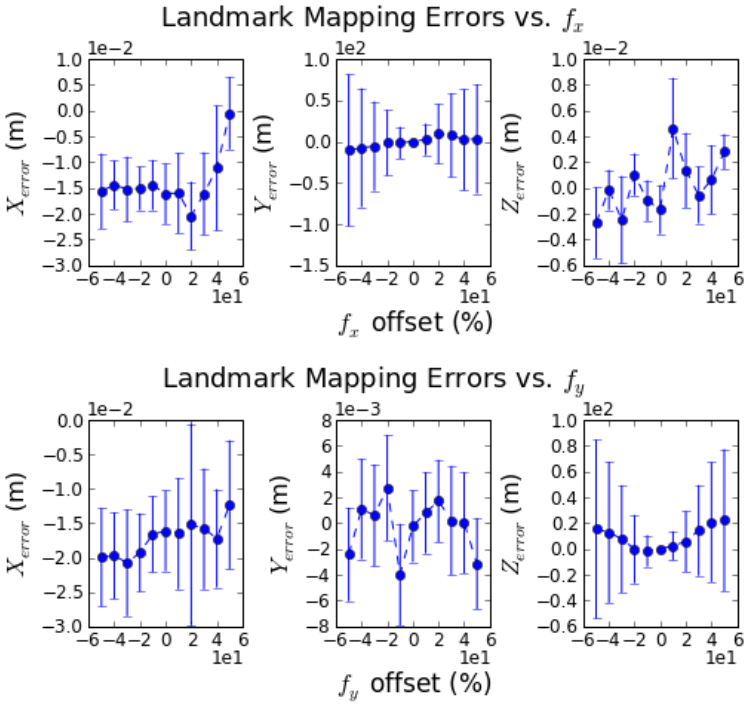

Figure 6.15: Landmark mapping error statistics with various (f_x, f_y) settings

also indicates that the mapping errors are dependent on landmark ground truth positions and error in (f_x, f_y). When f_x contains errors, landmark position errors on Y are directly proportional to the Y component of their ground truth positions, with the error in f_x determining the steepness of the slope. The same relation can be found for Z_{error} with f_y, and $Z_{\text{ground truth}}$.

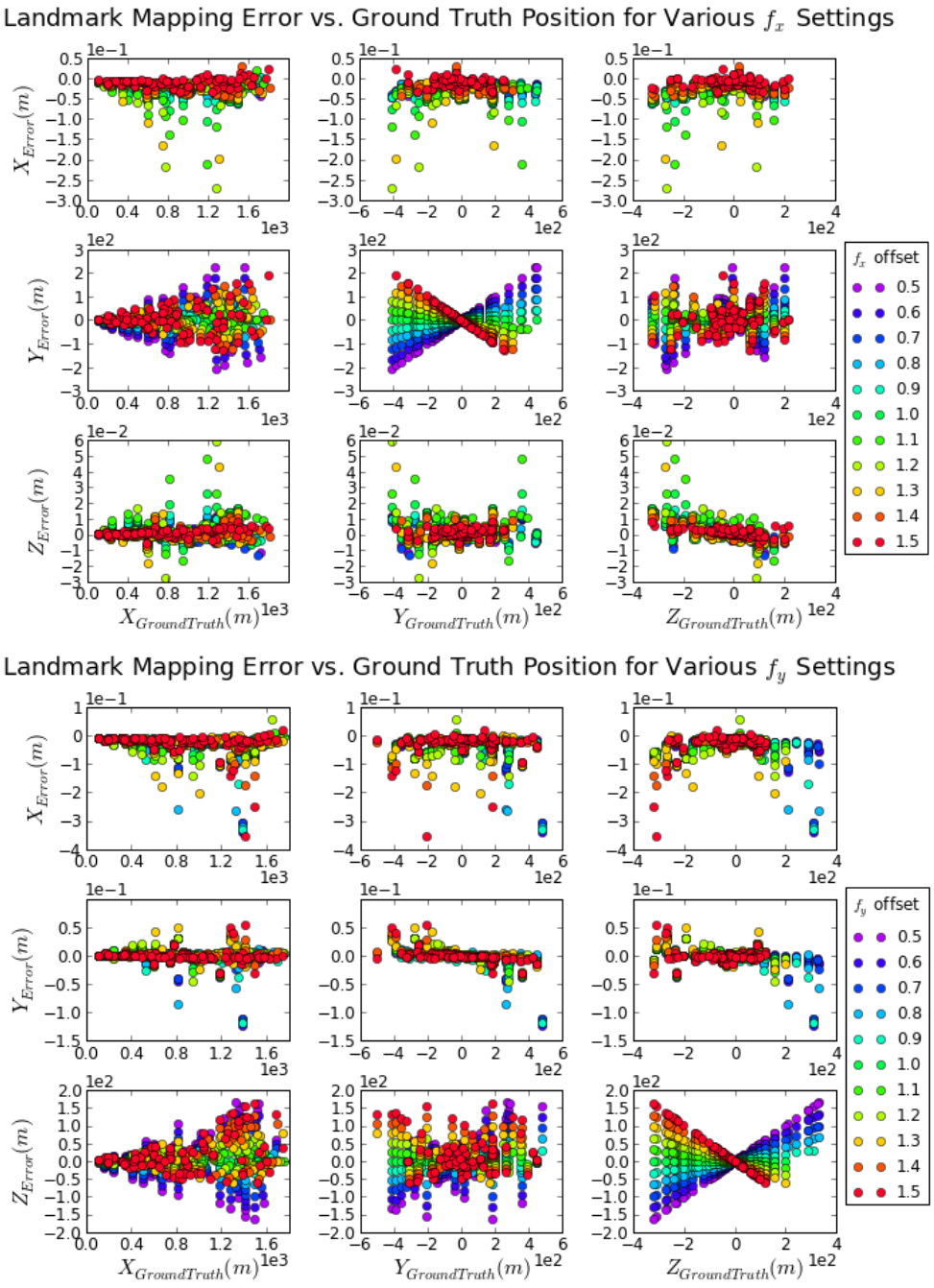

Figure 6.16: Landmark mapping errors versus ground truth landmark positions for various (f_x, f_y) settings

6.3.3 Effect of Errors in Lens Distortion

The CC-EKF-SLAM algorithm does not consider camera lens distortion at this stage. Therefore, the simulation evaluated the effect of lens distortion varying from 0% to 120% of the base value to estimate the amount of error resulting from ignoring the lens distortion.

Figure 6.17 shows the UAV localization error for various lens distortion settings. Ignoring the distortion brings significant error into the UAV localization. UAV position on the X axis receives the most impact. The errors are up to 100 meters, and have increasing standard deviation. Positions on the Y and Z axes show less error, but standard deviation grows larger with the increasing amount of distortion added.

Figure 6.18 reveals the cause of increasing standard deviation. UAV position errors are diverging in time and with the amount of lens distortion added. UAV orientation errors increase from maximum error of 5e-6 rad in low noise simulation to 2.5e-4 rad. Similar to position error, the plots of orientation error vs. frame number shows that the error amplitude increases with time but fluctuates around zero.

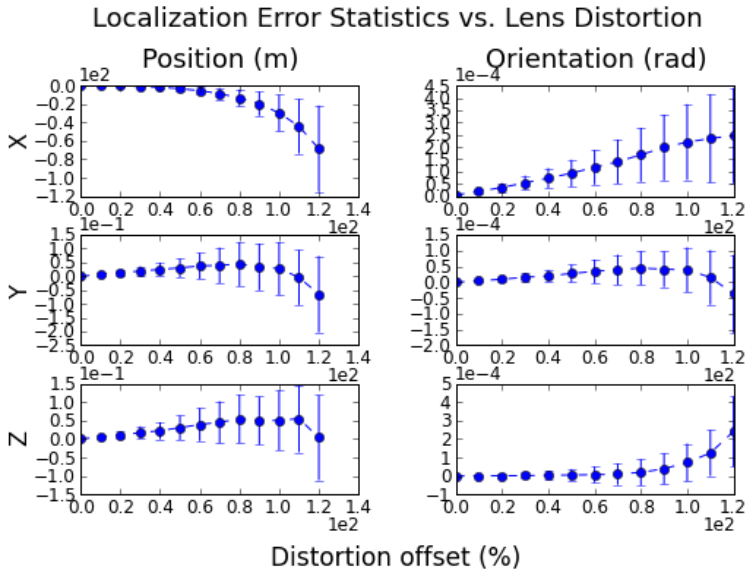

Figure 6.17: UAV localization error statistics with various lens distortion setting

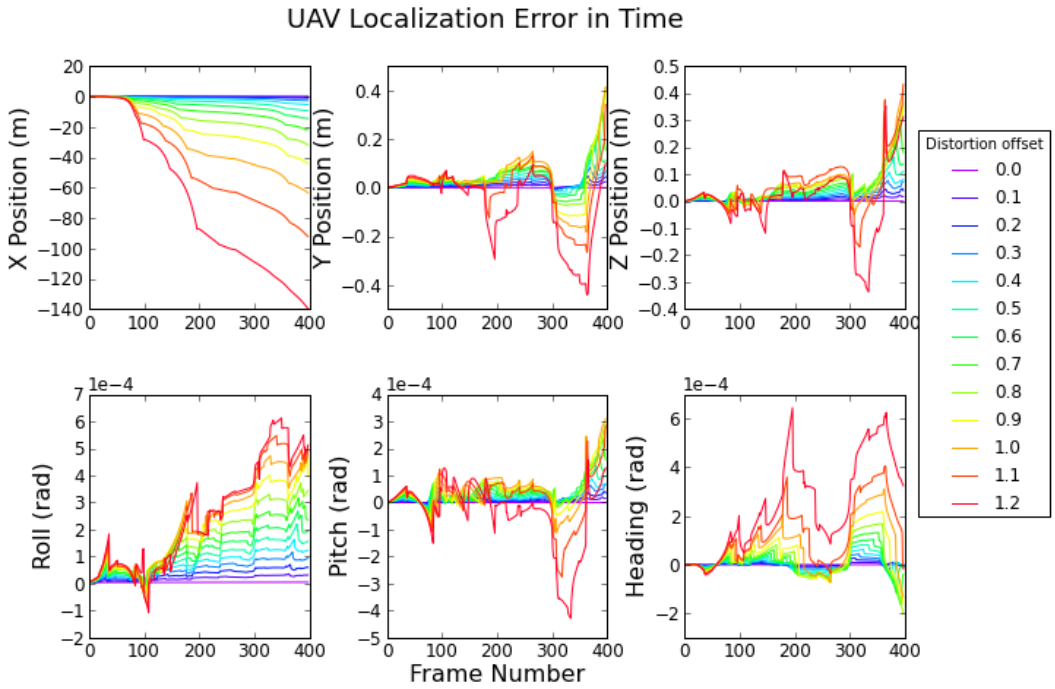

Figure 6.18: Diverging UAV localization errors for various lens distortion settings

The landmark position error statistics are shown in Figure 6.19. The position errors are also plotted against landmark ground truth positions in Figure 6.20. The

landmark positions on the X axis show the most error. The mean error reaches 400 meters with distortion offset at 120%. The amount of error is related to the landmarks' distance from the X axis which aligns with the optical axis (Figure 6.20, subplots [1,2] and [1,3]). The further the landmarks lay from the X axis, the bigger the errors they suffer. Landmark positions on the Y and Z axes have less mean error, but the standard deviation are bigger. Figure 6.20 subplots [2,2] and [3,3] show that the errors on the Y (or Z) axis are loosely proportional to the landmark ground truth coordinates on the Y (or Z) axis, where the amount of error in distortion determines the slope of the line. The relation indicates that landmarks lying around the edges of the image plane carry more errors than landmarks at the center of the image on all axes. This is exactly the kind of error that radial distortion corrects for.

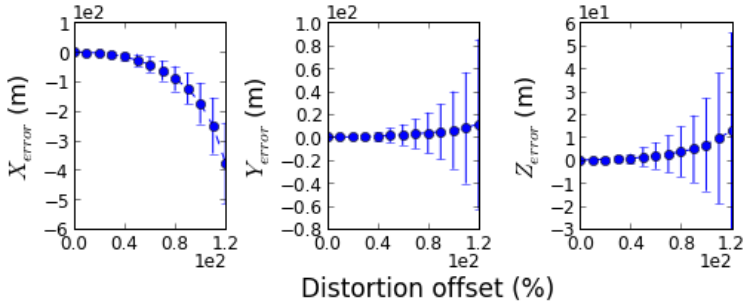

Figure 6.19: Landmark mapping error statistics with various lens distortion settings

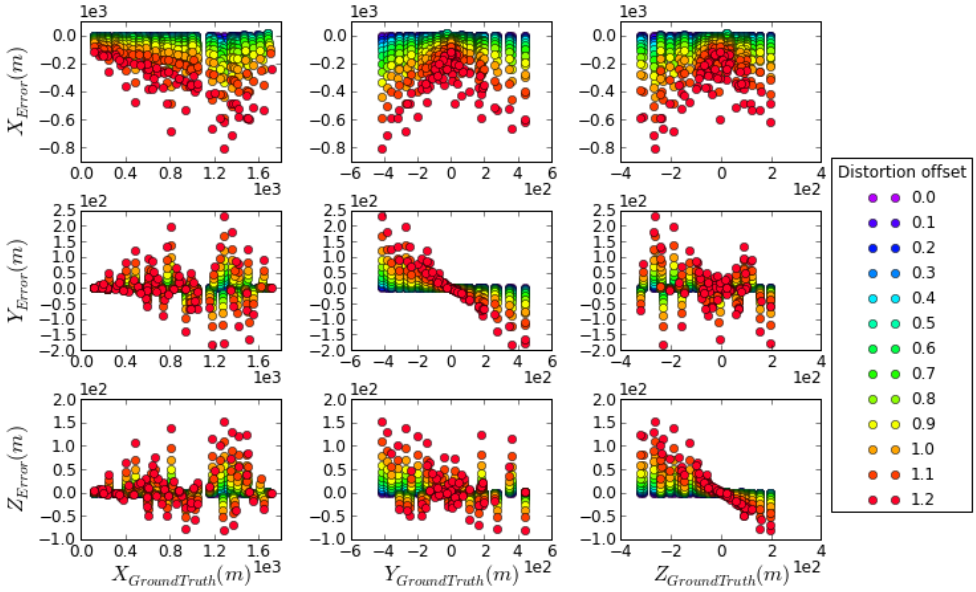

Figure 6.20: Landmark mapping errors versus landmark distance to optical axis with error in lens distortion

6.3.4 Effect of Image Digitization

It is well known that higher resolution sensors give more accuracy to the estimates. To know how high a resolution is good enough for the distance that this research is targeting for, a quantitative analysis is necessary. Simulations were run with various image resolution settings, and the results are shown in Figure 6.21.

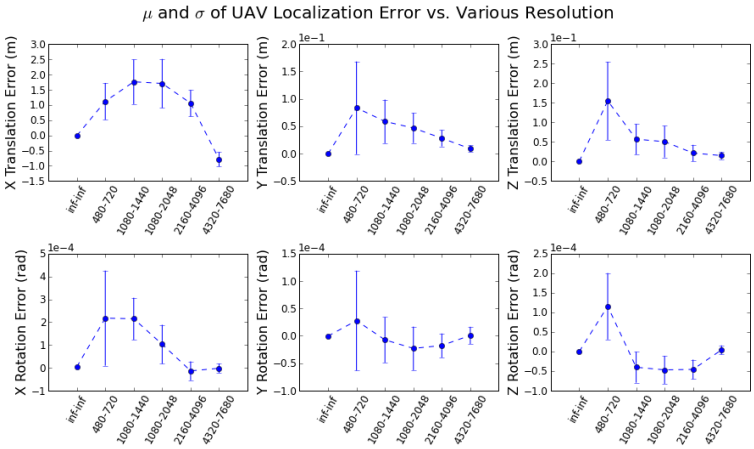

Figure 6.21: UAV localization error statistics for various image resolutions

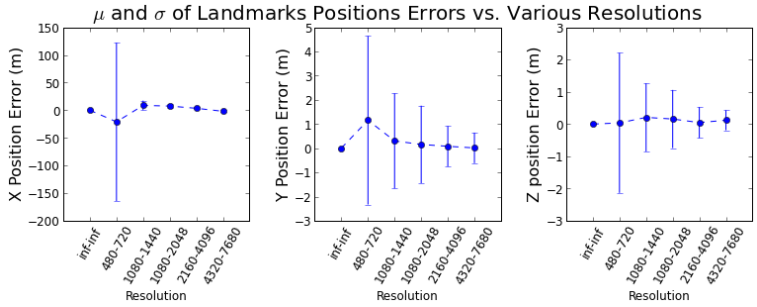

Figure 6.22: Landmark mapping error statistics for various image resolutions

This test confirms that the higher resolution the image sensor has, the more accuracy it will bring. The most significant error is seen at resolution 480x720 where the landmark position errors on X are +/- 150m. At 1080x1440, landmark position errors on the X axis are greatly reduced to a few meters. For all other parameter estimates, improvements at each increment of resolution are nearly linear. This result suggests that, to achieve reasonably good accuracy for obstacle detection, a sensor with resolution of 1080x1440 or higher is preferred.

Chapter 7

Conclusion

This thesis describes a research that addresses the obstacle detection problem for a low flying UAV using a monocular camera. The research focused on static obstacle detection for a medium size UAV deployed in a natural environment and performing a low altitude terrain following flight. An EKF based SLAM algorithm CC-EKF-SLAM was proposed in Chapter 4. The algorithm utilizes multiple sensors and inverse depth parametrization to detect long distance objects that range up to a thousand meters. A flight test was performed to collect aerial video and sensor recordings, accomplished by towing a sensor-loaded SUAS with a helicopter. Two pieces of aerial video and data were processed by the CC-EKF-SLAM algorithm. Convergence and consistency characteristics of the algorithm were studied. In addition, to evaluate the accuracy of the results, a digital terrain map of the survey zone was processed to compare to the results from the algorithm. Further analysis of the algorithm and error sources was performed through a series of simulations which simulated UAV motion as seen in flight and camera calibration errors. Both flight and simulation results are summarized in this chapter. Suggestions on future work are also described at the end.

7.1 Result Summaries

7.1.1 Test Results from Real Flight Data

The flight data was processed, and results were discussed in Chapter 5. Sparse corner features were extracted from the flight image sequence, and were tracked by the CC-EKF-SLAM algorithm to estimate landmark positions. An estimated terrain map and the trajectory of the SUAS were generated from the video sequence. The estimated terrain map was compared to the downloaded DEM, and the SUAS trajectory was compared to the on-board GPS and magnetometer recording. Results from the flight data are as follows:

- For SUAS localization accuracy, position error accumulates gradually, and reaches a maximum of 20 meters on the X axis, 50 meters on the Y axis and 30 meters on the Z axis at the end of the video sequence. The error for orientation estimates are within 1.15° for all three components.

- For landmark mapping, most landmarks tracked by the algorithm have their inverse depths converged to stable values, with the furthest landmark located at 1600 meters from SUAS location at the first frame.

- The accuracy of the estimated landmark locations achieve an average error of -56 meters on the X axis, 13 meters on the Y axis and 1.3 meters on the Z axis.

- Landmarks added after the first frame show offset errors on the Y axis. This is due to the error in the SUAS localization and the propagation of this error into the landmark mapping through reference frame transformation from the camera frame to the world frame.

Due to the difficulty in matching the tracked landmarks with the DEM in natural scene, an extra video was processed with the airport buildings in the scenes which

makes manual visual correspondence possible. All landmarks were matched manually to the satellite image from Google Earth. In this test, landmark positions had errors of 150 meters on the X axis, 40 meters on the Y axis and 20 meters on the Z axis. Given that they are all located at the corner of the image plane, this error is likely due to lens distortion.

7.1.2 Noise Analysis through Simulation

To better understand noise source in the CC-EKF-SLAM algorithm, a series of simulations were done with variable settings for the camera motions, camera intrinsic parameters, and image resolutions. A nearly no noise environment was first simulated with the UAV moving forward only. Effects from various UAV motions were simulated next. Then various camera model mismatch and image resolution settings were simulated. The discussion of the simulation results was presented in Chapter 6.

Low noise environment

Under forward only motion, UAV position errors are less than 1cm, and orientation errors are under $3e-3^{\circ}$. Landmark position errors are under 0.2 meters on the X axis, and under 0.02 meters on the Y and Z axes. The result shows error introduced by the algorithm itself under simple forward motion was very minor.

Impact from motion

The effects of oscillatory UAV motions were analyzed next. The UAV remained forward motion while oscillatory motion on each of the other 5 DOFs was added one at a time. The added motion was simulated with a 1Hz Sine wave and with variable amplitude settings. Results show that translational motion on the Y axis and the Z

axis and rotation around X axis (roll) have little effect on the UAV localization accuracy. Rotation around the Y axis (pitch) and rotation around the Z axis (heading) cause UAV positioning errors up to 30 meters peak-to-peak on Z and Y respectively, depending on the amplitude of the rotation. For landmark mapping accuracy, translation motion only increase landmark position errors by centimeters; rotational motion increase landmark position errors by meters for rotation around X, and hundreds of meters for rotation around Y and Z. Most errors appear on landmarks added after the first frame, caused by an initialization offset on parameter φ^W originating from the errors in UAV localization.

Impact from error in camera calibration

The second part of Chapter 6 analyzed the impact from the error in camera calibration. Results are summarized below:

Effect of Error in (c_x, c_y)

- Calibration error in c_x or c_y causes UAV position estimates to diverge from the correct value with time.
- Calibration error in c_x or c_y causes errors in landmark position estimates on all three axes. c_x affects the X and Y components, and c_y affects the X and Z components. The amount of error is related to the locations of the landmarks, and the amount of error in c_x or c_y through the first order polynomial equations.

Effect of Error in (f_x, f_y)

- Calibration error in f_x or f_y has little effect on UAV localization estimates.
- Calibration error in f_x affects the landmark positions on the Y component, error in f_y affects the Z component. The amount of error in landmark position

estimates are dependent on the actual landmark locations and the amount of error in f_x or f_y through the first order polynomial equations.

Effect of Distortion

- Ignoring distortion brings errors into all six parameters of UAV localization. The position error on the X axis is diverging from the true value with time. All other parameters oscillate around zero, but the amount of error grows bigger with time.
- Errors in landmark mapping resulted from ignoring the distortion are related to the landmark positions on the Y and Z axes. The further the landmark is positioned from the optical center, the greater the error becomes.

Effect of Image Resolution

- To achieve good accuracy on both self localization and landmark mapping, higher image resolution is a must.
- The 480x720 resolution used in the flight test could result in landmark mapping errors of up to +/-150 meters on the X axis.
- Increasing resolution to 1080x1440 significantly reduces the variance of errors on both localization and mapping estimates.

7.2 Future Research

This research has presented a SLAM framework for obstacle detection through a monocular camera. Flight test proved the feasibility of the algorithm for mapping landmarks more than 1 thousand meters away. To make the algorithm practical in a

realistic survey environment, a number of improvements can be made to increase its accuracy and reliability.

1. Including lens distortion into the measurement model should improve the accuracy of the algorithm, as suggested by the error analysis.

2. Sensor resolution should be increased if a future flight test opportunity arises.

3. A filter for checking the quality of corner features should be added. Landmarks that have poor corner signatures lead to greater tracking errors. In addition, due to the accumulated tracking error, the visual pattern of a landmark can be completely different from what it was when first extracted. Since there are correlations between landmarks, these tracking errors have an impact on other landmarks and UAV localization estimates. Removing the bad landmarks, and replacing them with good quality ones is necessary to make the algorithm more robust.

4. To enable large area mapping, a map joining algorithm should be added.

5. Localization error has a big impact on the accuracy of the map. To minimize localization error, when GPS data is available, which is the case most of the time, all parameters should be synchronized to GPS periodically. This procedure should also help to prevent consistency problems. GPS syncing could be integrated with the map joining algorithm.

Besides the improvements listed above, the cause of the sensitivity of the algorithm to rotation on the Y and Z axes should be studied further, and a method of compensating for the error should be developed.

List of References

[1] A. Nemra and N. Aouf, "Robust cooperative UAV visual SLAM," in *Proceedings of the 9th IEEE International Conference on Cybernetic Intelligent Systems*, 2010, pp. 1–6.

[2] S. Jianli, P. Shuang, W. Yugiang, and W. Xibin, "Unscented FastSLAM for UAV," in *Proceedings of the International Conference on Computer Science and Network Technology*, vol. 4, 2011, pp. 2529–2532.

[3] N. Sunderhauf, S. Lange, and P. Protzel, "Using the unscented Kalman filter in mono-SLAM with inverse depth parametrization for autonomous airship control," in *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, 2007, pp. 1–6.

[4] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragn, C. Martnez, and M. Olivares, "Visual 3-D SLAM from UAVs," *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4-5, pp. 299–321, Aug. 2009.

[5] D. Smith and S. Singh, "Approaches to multisensor data fusion in target tracking: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 12, pp. 1696–1710, 2006.

[6] T. James, "GeoSurv II UAV system requirements document," Carleton University, Tech. Rep., Mar. 2008.

[7] F. Zhang, R. Goubran, and P. Straznicky, "Obstacle detection for low flying UAS using monocular camera," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2012, pp. 2133–2137.

[8] M. C. Lu, C. C. Hsu, and Y. Y. Lu, "Distance and angle measurement of distant objects on an oblique plane based on pixel variation of CCD image," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2010, pp. 318–322.

[9] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo based obstacle detection for an unmanned air vehicle," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2006, pp. 2830–2835.

[10] P. Pinies, T. Lupton, S. Sukkarieh, and J. Tardos, "Inertial aiding of inverse depth SLAM using a monocular camera," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2007, pp. 2797–2802.

[11] Y. Li, Q. Pan, C. Zhao, and F. Yang, "Scene matching based EKF-SLAM visual navigation," in *Proceedings of the 2012 31st Chinese Control Conference*, 2012, pp. 5094–5099.

[12] A. De Angelis, M. Dionigi, A. Moschitta, and P. Carbone, "A low-cost ultra-wideband indoor ranging technique," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2007, pp. 1–6.

[13] F. Alonge, M. Branciforte, and F. Motta, "A novel method of distance measurement based on pulse position modulation and synchronization of chaotic signals using ultrasonic radar systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 2, pp. 318–329, 2009.

[14] M. Harb, R. Abielmona, K. Naji, and E. Petriu, "Neural networks for environmental recognition and navigation of a mobile robot," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2008, pp. 1123–1128.

[15] M. M. Saad, C. J. Bleakley, and S. Dobson, "Robust high-accuracy ultrasonic range measurement system," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 10, pp. 3334–3341, 2011.

[16] S. B. Williams, "Efficient solutions to autonomous mapping and navigation problems," Ph.D. dissertation, University of Sydney, 2001.

[17] K. S. Chong and L. Kleeman, "Feature-based mapping in real, large scale environments using an ultrasonic array," *International Journal of Robotics Research*, vol. 18, no. 1, pp. 3–19, 1999.

[18] E. Hanna, P. Straznicky, and R. Goubran, "Obstacle detection for low flying unmanned aerial vehicles using stereoscopic imaging," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, 2008, pp. 113–118.

[19] J. Civera, A. Davison, and J. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[20] R. Jirawimut, S. Prakoonwit, F. Cecelja, and W. Balachandran, "Visual odometer for pedestrian navigation," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 4, pp. 1166–1173, 2003.

[21] F. Amzajerdian, D. Pierrottet, L. Petway, G. Hines, and V. Roback, "Lidar systems for precision navigation and safe landing on planetary bodies," in *Proceedings of SPIE 8192, International Symposium on Photoelectronic Detection and Imaging*, vol. 819202, 2011.

[22] M. Nieuwenhuisen, D. Droeschel, M. Beul, and S. Behnke, "Obstacle detection and navigation planning for autonomous micro aerial vehicles," in *Proceedings of the 2014 International Conference on Unmanned Aircraft Systems*, May 2014, pp. 1040–1047.

[23] R. Subharsanan and R. Moss, "Low cost scanning LiDAR imager," Mar. 2013. [Online]. Available: http://www.lidarnews.com/PDF/LiDARMagazine_SudharsananMoss-LowCostLiDARImager_Vol3No2.pdf

[24] M. Lemmens, "Airborne LiDAR sensors," Feb. 2007. [Online]. Available: http://gim-international.com/files/productsurvey_v_pdfdocument_11.pdf

[25] E. Einhorn, C. Schrter, and H. M. Gross, "Can't take my eye off you: Attention-driven monocular obstacle detection and 3D mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 816–821.

[26] H. Hashimoto, T. Yamaura, and M. Higashiguchi, "Detection of obstacle from monocular vision based on histogram matching method," in *Proceedings of the 22nd IEEE International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 2, 1996, pp. 1047–1051.

[27] K. Yamaguchi, T. Kato, and Y. Ninomiya, "Moving obstacle detection using monocular vision," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2006, pp. 288–293.

[28] D. Maier, M. Bennewitz, and C. Stachniss, "Self-supervised obstacle detection for

humanoid navigation using monocular vision and sparse laser data," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 1263–1269.

[29] S. Kubota, T. Nakano, and Y. Okamoto, "A global optimization algorithm for real-time on-board stereo obstacle detection systems," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2007, pp. 7–12.

[30] Y. Xu, M. Zhao, X. Wang, Y. Zhang, Y. Peng, Y. Yuan, and H. Liu, "A method of stereo obstacle detection based on image symmetrical move," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 36–41.

[31] Z. Zhang, Y. Wang, J. Brand, and N. Dahnoun, "Real-time obstacle detection based on stereo vision for automotive applications," in *Proceedings of the 5th European DSP Education and Research Conference*, 2012, pp. 281–285.

[32] W. Van der Mark, J. Van Den Heuvel, and F. C. A. Groen, "Stereo based obstacle detection with uncertainty in rough terrain," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2007, pp. 1005–1012.

[33] A. Broggi, M. Buzzoni, M. Felisa, and P. Zani, "Stereo obstacle detection in challenging environments: The VIAC experience," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1599–1604.

[34] F. Plumet, C. Petres, M.-A. Romero-Ramirez, B. Gas, and S.-H. Ieng, "Toward an autonomous sailing boat," *IEEE Journal of Oceanic Engineering*, vol. PP, no. 99, pp. 1–11, 2014.

[35] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., Oct. 2008.

[36] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.

[37] R. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, pp. 364–374.

[38] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[39] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[40] H. Sorenson, "Least-squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, no. 7, pp. 63–68, 1970.

[41] Analytic Sciences Corporation, *Applied Optimal Estimation.* Cambridge, Mass: M.I.T. Press, 1974.

[42] M. S. Grewal, *Kalman Filtering: Theory and Practice.* Englewood Cliffs, N.J: Prentice-Hall, 1993.

[43] F. L. Lewis, *Optimal Estimation: with an Introduction to Stochastic Control Theory.* New York: Wiley, 1986.

[44] R. G. Brown, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed. New York: John Wiley & Sons, Inc., 1993.

[45] P. S. Maybeck, *Stochastic Models, Estimation and Control.* New York: Academic Press, 1979, no. v.141.

[46] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[47] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[48] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, Dec. 1986.

[49] H. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal of Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[50] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1991, pp. 1442–1447 vol.3.

[51] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles," in *Robotics Research*. Springer London, Jan. 1996, pp. 613–625.

[52] M. Csorba, J. K. Uhlmann, and H. F. Durrant-Whyte, "New approach to simultaneous localization and dynamic map building," in *Proceeding of the SPIE 2738, Navigation and Control Technologies for Unmanned Systems*, vol. 2738, 1996, pp. 26–36.

[53] M. Csorba, "Simultaneous localisation and map building," Ph.D. dissertation, University of Oxford, 1997.

[54] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[55] A. Martinelli, N. Tomatis, and R. Siegwart, "Some results on SLAM and the closing the loop problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2917–2922.

[56] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, 2001, pp. 4238–4243.

[57] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3562–3568.

[58] U. Frese, "A discussion of simultaneous localization and mapping," *Autonomous Robots*, vol. 20, no. 1, pp. 25–42, Jan. 2006.

[59] J. A. Castellanos, J. Neira, and J. D. Tards, "Limits to the consistency of EKF-based SLAM," in *Proceedings of the 5th IFAC Symposium Intelligent Autonomous Vehicles*, 2004.

[60] G. P. Huang, A. Mourikis, and S. Roumeliotis, "Analysis and improvement of the consistency of extended Kalman filter based SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008, pp. 473–479.

[61] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended Kalman filter based SLAM," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[62] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point RANSAC for EKF-based structure from motion," in *Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems*, 2009, pp. 3498–3504.

[63] C. Estrada, J. Neira, and J. Tardos, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[64] J. Leonard and P. Newman, "Consistent, convergent, and constant-time SLAM," in *Proceedings of the 18th international joint conference on Artificial intelligence*, 2003, pp. 1143–1150.

[65] M. Bosse, P. Newman, J. Leonard, and S. Teller, "SLAM in large-scale cyclic environments using the Atlas framework," *International Journal of Robotics Research*, pp. 1113–1139, 2004.

[66] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," Ph.D. dissertation, University of Sydney, 2002.

[67] M. Okutomi and T. Kanade, "A multiple-baseline stereo," in *Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991, pp. 63–69.

[68] A. Jepson and D. Heeger, "A fast subspace algorithm for recovering rigid motion," in *Proceedings of the IEEE Workshop on Visual Motion*, 1991, pp. 124–131.

[69] A. K. R. Chowdhury and R. Chellappa, "Stochastic approximation and rate-distortion analysis for robust structure and motion estimation," *International Journal of Computer Vision*, vol. 55, no. 1, pp. 27–53, Oct. 2003.

[70] V. Aidala and S. Hammel, "Utilization of modified polar coordinates for bearings-only tracking," *IEEE Transactions on Automatic Control*, vol. 28, no. 3, pp. 283–294, 1983.

[71] "Athena 111m integrated flight control system." [Online]. Available: http://www.rockwellcollins.com

[72] "CGIAR-CSI SRTM 90m DEM digital elevation database." [Online]. Available: http://srtm.csi.cgiar.org/

[73] "Python." [Online]. Available: http://www.python.org/

[74] "OpenCV." [Online]. Available: http://opencv.org/

[75] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[76] J. Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker, description of the algorithm," *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, 1999.

[77] "World geodetic system - Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/World_Geodetic_System

[78] "pyproj - Python interface to PROJ.4 library." [Online]. Available: http://code.google.com/p/pyproj/

[79] "GPS accuracy." [Online]. Available: http://www.gps.gov/systems/gps/performance/accuracy/

[80] "Google Earth." [Online]. Available: http://www.google.com/earth/

Appendix A

Coordinate Transformation

For a mobile robot, a point $\boldsymbol{P}$ in space has coordinates $[x \quad y \quad z]$ in the world frame, and $[x' \quad y' \quad z']$ in the mobile frame, as illustrated in Figure A.1

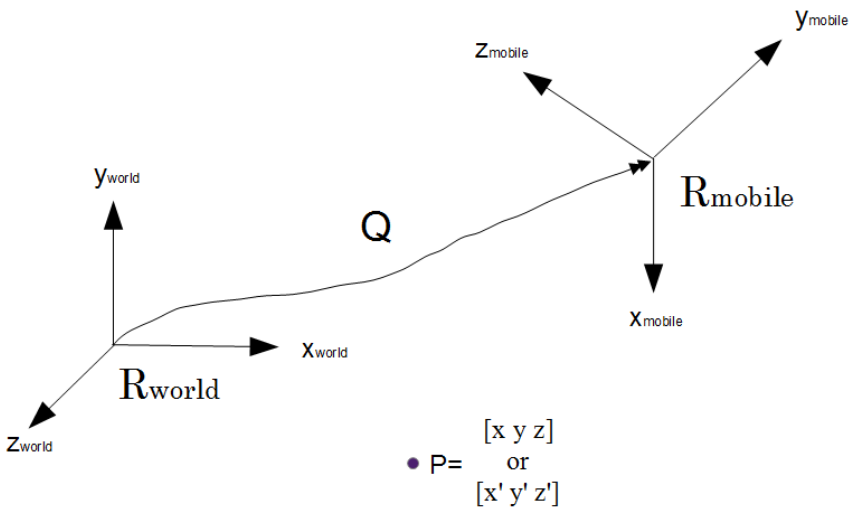

Figure A.1: A point in the world frame and the mobile frame

The coordinates for the same point $\boldsymbol{P}$ in the world frame and the mobile frame

are related by

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{World} = \boldsymbol{Q} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}_{mobile} \tag{A.1}$$

where $\boldsymbol{Q}$ is the transformation matrix composed of rotation and translation that evolve the world frame to the mobile frame. The calculation of $\boldsymbol{Q}$ is given by

$$\boldsymbol{Q} = \boldsymbol{Q_T} \boldsymbol{Q_{R_z}} \boldsymbol{Q_{R_y}} \boldsymbol{Q_{R_x}} \tag{A.2}$$

$\boldsymbol{Q_T}$, $\boldsymbol{Q_{R_x}}$, $\boldsymbol{Q_{R_y}}$, $\boldsymbol{Q_{R_z}}$ are the translation and rotation matrices for axes X, Y and Z. They are give in Equation (A.3) to (A.6).

$$\boldsymbol{Q_T} = \begin{bmatrix} 1 & 0 & 0 & x_T \\ 0 & 1 & 0 & y_T \\ 0 & 0 & 1 & z_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$\boldsymbol{Q_{R_x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(r_x) & -\sin(r_x) & 0 \\ 0 & \sin(r_x) & \cos(r_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$\boldsymbol{Q_{R_y}} = \begin{bmatrix} \cos(r_y) & 0 & \sin(r_y) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(r_y) & 0 & \cos(r_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

$$\boldsymbol{Q_{R_z}} = \begin{bmatrix} \cos(r_z) & -\sin(r_z) & 0 & 0 \\ \sin(r_z) & \cos(r_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

The forward transformation calculates point $\boldsymbol{P}$'s coordinates in the world frame given its location in the mobile frame. To calculate point $\boldsymbol{P}$'s coordinates in the mobile frame from its location in the world frame, inverse transformation can be used. The inverse transformation matrix is given in Equation (A.7) - (A.11)

$$\boldsymbol{Q^{-1}} = \boldsymbol{Q_{R_x}^{-1}} \boldsymbol{Q_{R_y}^{-1}} \boldsymbol{Q_{R_z}^{-1}} \boldsymbol{Q_T^{-1}} \tag{A.7}$$

$$\boldsymbol{Q_T^{-1}} = \begin{bmatrix} 1 & 0 & 0 & -x_T \\ 0 & 1 & 0 & -y_T \\ 0 & 0 & 1 & -z_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

$$\boldsymbol{Q}_{\boldsymbol{R_x}}^{-1} = \boldsymbol{Q}_{\boldsymbol{R_x}}^{T} \tag{A.9}$$

$$\boldsymbol{Q}_{\boldsymbol{R_y}}^{-1} = \boldsymbol{Q}_{\boldsymbol{R_y}}^{T} \tag{A.10}$$

$$\boldsymbol{Q}_{\boldsymbol{R_z}}^{-1} = \boldsymbol{Q}_{\boldsymbol{R_z}}^{T} \tag{A.11}$$

Appendix B

Jacobian Matrix for Filter Initialization Equations

The Jacobian matrix used in the state covariance matrix initialization Equation (4.23) is derived in this section. The complete Jacobian matrix $\boldsymbol{J}$ for initializing a landmark covariance is given by

$$\boldsymbol{J} = \begin{bmatrix} & & & & & & \mathbf{0} \\ & & & \boldsymbol{I} & & & \vdots \\ & & & & & & \mathbf{0} \\ \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{OX_W^C}} & \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{c^C}} & \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{r^C}} & \mathbf{0} & \dots & \mathbf{0} & \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{g_i}} \end{bmatrix} \tag{B.1}$$

Whenever a new landmark is added, its initial position is always $[0 \quad 0 \quad 0]$ in the camera frame, and the $[\rho \quad \varphi \quad \theta]$ parameters are not a function of $\boldsymbol{OX_W^C}$, $\boldsymbol{c^C}$,or $\boldsymbol{r^C}$, therefore

$$\frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{OX_W^C}} = \mathbf{0_{6\times 6}}, \quad \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{c^C}} = \mathbf{0_{6\times 3}}, \quad \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{r^C}} = \mathbf{0_{6\times 3}} \tag{B.2}$$

$\boldsymbol{J}$ can then be simplified as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\boldsymbol{\partial p_i}}{\boldsymbol{\partial g_i}} \end{bmatrix} \tag{B.3}$$

where $\boldsymbol{g_i}$ includes the variable in matrix $\boldsymbol{R}$ given in Equation(4.24): $\boldsymbol{g_i} = [x_i^C \quad y_i^C \quad z_i^C \quad \rho_i \quad u_i \quad v_i]$. Then

$$\frac{\boldsymbol{\partial p_i}}{\boldsymbol{\partial g_i}} = \begin{bmatrix} \boldsymbol{I_{3\times 3}} & & \boldsymbol{0_{3\times 3}} & \\ & \frac{\partial \rho_i}{\partial \rho_i} & \frac{\partial \rho_i}{\partial u_i} & \frac{\partial \rho_i}{\partial v_i} \\ \boldsymbol{0_{3\times 3}} & \frac{\partial \varphi_i^C}{\partial \rho_i} & \frac{\partial \varphi_i^C}{\partial u_i} & \frac{\partial \varphi_i^C}{\partial v_i} \\ & \frac{\partial \theta_i^C}{\partial \rho_i} & \frac{\partial \theta_i^C}{\partial u_i} & \frac{\partial \theta_i^C}{\partial v_i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I_{3\times 3}} & & \boldsymbol{0_{3\times 3}} & \\ & 1 & 0 & 0 \\ \boldsymbol{0_{3\times 3}} & 0 & \frac{\partial \varphi_i^C}{\partial u_i} & \frac{\partial \varphi_i^C}{\partial v_i} \\ & 0 & \frac{\partial \theta_i^C}{\partial u_i} & \frac{\partial \theta_i^C}{\partial v_i} \end{bmatrix} \tag{B.4}$$

Based on the chain rule of differentiation, the partial derivatives in Equation B.4 is given by

$$\frac{\partial \varphi_i^C}{\partial u_i} = \frac{\boldsymbol{\partial \varphi_i^C}}{\boldsymbol{\partial h^C}} \frac{\boldsymbol{\partial h^C}}{\boldsymbol{\partial u_i}} \tag{B.5}$$

$$\frac{\partial \theta_i^C}{\partial u_i} = \frac{\boldsymbol{\partial \theta_i^C}}{\boldsymbol{\partial h^C}} \frac{\boldsymbol{\partial h^C}}{\boldsymbol{\partial u_i}} \tag{B.6}$$

$$\frac{\partial \varphi_i^C}{\partial v_i} = \frac{\boldsymbol{\partial \varphi_i^C}}{\boldsymbol{\partial h^C}} \frac{\boldsymbol{\partial h^C}}{\boldsymbol{\partial v_i}} \tag{B.7}$$

$$\frac{\partial \theta_i^C}{\partial v_i} = \frac{\boldsymbol{\partial \theta_i^C}}{\boldsymbol{\partial h^C}} \frac{\boldsymbol{\partial h^C}}{\boldsymbol{\partial v_i}} \tag{B.8}$$

When lens distortion is ignored, Equation B.5-B.8 can be calculated using Equation

B.9 through B.12.

$$\frac{\partial \boldsymbol{\varphi_i^C}}{\partial \boldsymbol{h^C}} = \begin{bmatrix} \frac{\partial \varphi_i^C}{\partial h_x^C} & \frac{\partial \varphi_i^C}{\partial h_y^C} & \frac{\partial \varphi_i^C}{\partial h_z^C} \end{bmatrix} = \begin{bmatrix} \frac{-h_x^C \cdot h_z^C}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})\sqrt{h_x^{C\,2}+h_y^{C\,2}}} \\ \frac{-h_y^W \cdot h_z^W}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})\sqrt{h_x^{C\,2}+h_y^{C\,2}}} \\ \frac{\sqrt{h_x^{C\,2}+h_y^{C\,2}}}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})} \end{bmatrix}^T \tag{B.9}$$

$$\frac{\partial \boldsymbol{\theta_i^C}}{\partial \boldsymbol{h^C}} = \begin{bmatrix} \frac{\partial \theta_i^C}{\partial h_x^C} & \frac{\partial \theta_i^C}{\partial h_y^C} & \frac{\partial \theta_i^C}{\partial h_z^C} \end{bmatrix} = \begin{bmatrix} -\frac{h_y^C}{h_x^{C\,2}+h_y^{C\,2}} & \frac{h_x^C}{h_x^{C\,2}+h_y^{C\,2}} & 0 \end{bmatrix} \tag{B.10}$$

$$\frac{\partial \boldsymbol{h^C}}{\partial \boldsymbol{u_i}} = \begin{bmatrix} \frac{\partial h_x^C}{\partial u_i} \\ \frac{\partial h_y^C}{\partial u_i} \\ \frac{\partial h_z^C}{\partial u_i} \end{bmatrix} = \begin{bmatrix} 0 \\ s_x \\ 0 \end{bmatrix} \tag{B.11}$$

$$\frac{\partial \boldsymbol{h^C}}{\partial \boldsymbol{v_i}} = \begin{bmatrix} \frac{\partial h_x^C}{\partial v_i} \\ \frac{\partial h_y^C}{\partial v_i} \\ \frac{\partial h_z^C}{\partial v_i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s_y \end{bmatrix} \tag{B.12}$$

Equations given above completes the Jacobian matrix calculation.

Appendix C

Linearization of Measurement Model and Composition Equations

The Jacobian matrix used in linearizing the measurement equations and composition equations are derived in this section.

C.1 Linearization of Measurement Model

The measurement model of the algorithm is not linear, and must be linearized by evaluating the first derivative of the measurement equation at the predicted camera location. Since the measurement model is not a function of the world origin coordinate and orientation, the Jacobian matrix of the measurement model is

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{k}}^{\boldsymbol{U}}(\boldsymbol{x})}{\partial \boldsymbol{x}_{\boldsymbol{k}}} = \begin{bmatrix} & \frac{\partial h_{1,k}^{U}(x)}{\partial c_k^C} & \frac{\partial h_{1,k}^{U}(x)}{\partial r_k^C} & \frac{\partial h_{1,k}^{U}(x)}{\partial p_{1,k}} & \cdots & \frac{\partial h_{1,k}^{U}(x)}{\partial p_{n,k}} \\ \mathbf{0_{2n\times 6}} & \vdots & \vdots & \vdots & \ddots & \vdots \\ & \frac{\partial h_{n,k}^{U}(x)}{\partial c_k^C} & \frac{\partial h_{n,k}^{U}(x)}{\partial r_k^C} & \frac{\partial h_{n,k}^{U}(x)}{\partial p_{1,k}} & \cdots & \frac{\partial h_{n,k}^{U}(x)}{\partial p_{n,k}} \end{bmatrix}$$

$$= \begin{bmatrix} & \frac{\partial h_{1,k}^{U}(x)}{\partial c_k^C} & \frac{\partial h_{1,k}^{U}(x)}{\partial r_k^C} & \frac{\partial h_{1,k}^{U}(x)}{\partial p_{1,k}} & \cdots & \mathbf{0} \\ \mathbf{0_{2n\times 6}} & \vdots & \vdots & \vdots & \ddots & \vdots \\ & \frac{\partial h_{n,k}^{U}(x)}{\partial c_k^C} & \frac{\partial h_{n,k}^{U}(x)}{\partial r_k^C} & \mathbf{0} & \cdots & \frac{\partial h_{n,k}^{U}(x)}{\partial p_{n,k}} \end{bmatrix} \tag{C.1}$$

Elements in the matrix above for landmark i at time step k are given in Equations (C.2)-(C.8). Subscript k is omitted.

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}(\boldsymbol{x})}{\partial \boldsymbol{c}^{\boldsymbol{C}}} = \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}}{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}} \cdot \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}}{\partial \boldsymbol{c}^{\boldsymbol{C}}} \tag{C.2}$$

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}(\boldsymbol{x})}{\partial \boldsymbol{r}^{\boldsymbol{C}}} = \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}}{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}} \cdot \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}}{\partial \boldsymbol{r}^{\boldsymbol{C}}} \tag{C.3}$$

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}(\boldsymbol{x})}{\partial \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{C}}} = \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}}{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}} \cdot \frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{C}}} \tag{C.4}$$

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{U}}}{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}} = \begin{bmatrix} -\frac{s_x h_{i,y}^C}{{h_{i,x}^C}^2} & \frac{s_x}{h_{i,x}^C} & 0 \\ -\frac{s_y h_{i,z}^C}{{h_{i,x}^C}^2} & 0 & \frac{s_y}{h_{i,x}^C} \end{bmatrix} \tag{C.5}$$

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{C}}}{\partial \boldsymbol{c}^{\boldsymbol{C}}} = -\boldsymbol{Q}^{-1}(\boldsymbol{r}^{\boldsymbol{C}})\rho_i \tag{C.6}$$

$$\frac{\partial \boldsymbol{h_i^C}}{\partial \boldsymbol{r^C}} = \begin{bmatrix} \frac{\partial \boldsymbol{h_i^C}}{\partial \boldsymbol{r_x^C}} & \frac{\partial \boldsymbol{h_i^C}}{\partial \boldsymbol{r_y^C}} & \frac{\partial \boldsymbol{h_i^C}}{\partial \boldsymbol{r_z^C}} \end{bmatrix} \tag{C.7}$$

$$\frac{\partial \boldsymbol{h_i^C}}{\partial \boldsymbol{p_i}} = \begin{bmatrix} \frac{\partial \boldsymbol{h_i^C}}{\partial x_i^C} & \frac{\partial \boldsymbol{h_i^C}}{\partial y_i^C} & \frac{\partial \boldsymbol{h_i^C}}{\partial z_i^C} & \frac{\partial \boldsymbol{h_i^C}}{\partial \rho_i} & \frac{\partial \boldsymbol{h_i^C}}{\partial \theta_i} & \frac{\partial \boldsymbol{h_i^C}}{\partial \varphi_i} \end{bmatrix} \tag{C.8}$$

where $\boldsymbol{Q^{-1}}$ can be found in Appendix A. Equation (C.8) is a 3x6 matrix

$$\begin{bmatrix} \frac{\partial \boldsymbol{h_i^C}}{\partial x_i^C} & \frac{\partial \boldsymbol{h_i^C}}{\partial y_i^C} & \frac{\partial \boldsymbol{h_i^C}}{\partial z_i^C} \end{bmatrix} = \boldsymbol{Q^{-1}(r^C)}\rho_i \tag{C.9}$$

$$\frac{\partial \boldsymbol{h_i^C}}{\partial \rho_i} = -\boldsymbol{Q^{-1}(r^C)} \cdot \boldsymbol{c} \tag{C.10}$$

$$\frac{\partial \boldsymbol{h_i^C}}{\partial \theta_i} = \boldsymbol{Q^{-1}(r^C)} \begin{bmatrix} -cos(\varphi_i)\sin(\theta_i) \\ \cos(\varphi_i)cos(\theta_i) \\ 0 \end{bmatrix} \tag{C.11}$$

$$\frac{\partial \boldsymbol{h_i^C}}{\partial \varphi_i} = \boldsymbol{Q^{-1}(r^C)} \begin{bmatrix} -cos(\varphi_i)sin(\theta_i) \\ -sin(\varphi_i)sin(\theta_i) \\ cos(\varphi_i) \end{bmatrix} \tag{C.12}$$

C.2 Jacobian Matrix of Composition Equations

The composition equations given in Chapter 4 are nonlinear. For the error matrix to propagate properly from one frame to another, the Jacobian matrix of the equations must be computed. The complete Jacobian matrix for the full state vector is given

by

$$\boldsymbol{J}_{\boldsymbol{C_{k-1}\to C_k}} = \frac{\boldsymbol{\partial x_k^{C_k}}}{\boldsymbol{\partial x_k^{C_{k-1}}}} = \begin{bmatrix} \frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{OX_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{p_{1,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{p_{2,k}^{C_{k-1}}}} & \cdots \end{bmatrix} \tag{C.13}$$

where superscript C_k refers to the new camera frame after composition, and superscript C_{k-1} refers to the camera frame before composition. $\frac{\partial \boldsymbol{x_k^{C_k}}}{\partial \boldsymbol{OX_{W,k}^{C_{k-1}}}}$ is given by:

$$\frac{\boldsymbol{\partial x_k^{C_k}}}{\boldsymbol{\partial OX_{W,k-1}^{C_{k-1}}}} = \begin{bmatrix} \frac{\partial \boldsymbol{O_{W,k}^{C_k}}}{\partial \boldsymbol{O_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{O_{W,k}^{C_k}}}{\partial \boldsymbol{W_{W,k}^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{W_{W,k}^{C_k}}}{\partial \boldsymbol{O_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{W_{W,k}^{C_k}}}{\partial \boldsymbol{W_{W,k}^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{c_k^{C_k}}}{\partial \boldsymbol{O_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{c_k^{C_k}}}{\partial \boldsymbol{W_{W,k}^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{r_k^{C_k}}}{\partial \boldsymbol{O_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{r_k^{C_k}}}{\partial \boldsymbol{W_{W,k}^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{p_i^{C_k}}}{\partial \boldsymbol{O_{W,k}^{C_{k-1}}}} & \frac{\partial \boldsymbol{p_i^{C_k}}}{\partial \boldsymbol{W_{W,k}^{C_{k-1}}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q^{-1}(r_k^{C_{k-1}})} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I_{3\times 3}} \\ \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{C.14}$$

The derivatives of $\boldsymbol{x_k^{C_k}}$ with respect to $\boldsymbol{c}$ and $\boldsymbol{r}$ are

$$\begin{bmatrix} \frac{\partial \boldsymbol{O_{W,k}^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{O_{W,k}^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{W_{W,k}^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{W_{W,k}^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{c_k^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{c_k^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{r_k^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{r_k^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{p(0:2)_i^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{p(0:2)_i^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \frac{\partial \boldsymbol{p(3:5)_i^{C_k}}}{\partial \boldsymbol{c_k^{C_{k-1}}}} & \frac{\partial \boldsymbol{p(3:5)_i^{C_k}}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{-Q^{-1}(r_k^{C_{k-1}})} & \frac{\partial \boldsymbol{Q^{-1}(r_k^{C_{k-1}})}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{-I_{3\times 3}} \\ \boldsymbol{I_{3\times 3}} & \boldsymbol{0_{3\times 3}} \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} \\ \boldsymbol{-Q^{-1}(r_k^{C_{k-1}})} & \frac{\partial \boldsymbol{Q^{-1}(r_k^{C_{k-1}})}}{\partial \boldsymbol{r_k^{C_{k-1}}}} \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{pr} \end{bmatrix} \tag{C.15}$$

where

$$\boldsymbol{pr} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{\partial \varphi_{i,k}^{C_k}}{\partial r_k^{C_{k-1}}} & & \\ \frac{\partial \theta_{i,k}^{C_k}}{\partial r_k^{C_{k-1}}} & & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & \frac{\partial m^{-1}(Q^{-1}(r_k^{C_{k-1}}) m(\varphi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}))}{\partial r_k^{C_{k-1}}} & \end{bmatrix} \tag{C.16}$$

Let $\boldsymbol{M} = \boldsymbol{Q^{-1}}(\boldsymbol{r_k^{C_{k-1}}})\boldsymbol{m}(\boldsymbol{\varphi_{i,k}^{C_{k-1}}}, \boldsymbol{\theta_{i,k}^{C_{k-1}}})$ represent the vector $\boldsymbol{m}$ seen by the camera at location k, and $\boldsymbol{m^{-1}} = \boldsymbol{m}^{-1}(\boldsymbol{M})$. Then

$$\frac{\boldsymbol{\partial m^{-1}}}{\boldsymbol{\partial r_k^{C_{k-1}}}} = \frac{\boldsymbol{\partial m^{-1}}}{\boldsymbol{\partial M}} \cdot \frac{\boldsymbol{\partial M}}{\boldsymbol{\partial r_k^{C_{k-1}}}} \tag{C.17}$$

where

$$\frac{\boldsymbol{\partial M}}{\boldsymbol{\partial r_k^{C_{k-1}}}} = \frac{\boldsymbol{\partial Q^{-1}}(\boldsymbol{r_k^{C_{k-1}}})}{\boldsymbol{\partial r_k^{C_{k-1}}}} \cdot \boldsymbol{m}(\boldsymbol{\varphi_{i,k}^{C_{k-1}}}, \boldsymbol{\theta_{i,k}^{C_{k-1}}}) \tag{C.18}$$

as $\boldsymbol{m}^{-1}$ is the function that calculates $[\varphi \quad \theta]$ from a given vector. $\frac{\boldsymbol{\partial m^{-1}}}{\boldsymbol{\partial M}}$ is the same as Equation B.9 and B.10.

The derivatives of $\boldsymbol{x_k^{C_k}}$ with respect to landmark parameters $\boldsymbol{p_{i,k}^{C_{k-1}}}$ are given in

$$\frac{\partial \boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} = \begin{bmatrix} \frac{\partial \boldsymbol{O}_{\boldsymbol{W,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{W}_{\boldsymbol{W,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{c}_{\boldsymbol{k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{r}_{\boldsymbol{k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \frac{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \end{bmatrix} \tag{C.19}$$

The last element is computed as

$$\frac{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} = \begin{bmatrix} \frac{\partial \boldsymbol{x}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{y}_{\boldsymbol{i,k}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{z}_{\boldsymbol{i,k}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{\rho}_{\boldsymbol{i,k}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{\varphi}_{\boldsymbol{i,k}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ \frac{\partial \boldsymbol{\theta}_{\boldsymbol{i,k}}^{\boldsymbol{C}}}{\partial \boldsymbol{p}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}^{-1}(\boldsymbol{r}_{\boldsymbol{k}}^{\boldsymbol{C_{k-1}}}) & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 3}} & \frac{\partial \boldsymbol{p}(\mathbf{3:5})_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}(\mathbf{3:5})_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \end{bmatrix} \tag{C.20}$$

where

$$\frac{\partial \boldsymbol{p}(\mathbf{3:5})_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{p}(\mathbf{3:5})_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\partial \varphi_{i,k}^{C_k}}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \varphi_{i,k}^{C_k}}{\partial \theta_{i,k}^{C_{k-1}}} \\ 0 & \frac{\partial \theta_{i,k}^{C_k}}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \theta_{i,k}^{C_k}}{\partial \theta_{i,k}^{C_{k-1}}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\partial \boldsymbol{\varphi}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\varphi}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} & \frac{\partial \boldsymbol{\varphi}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\theta}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \\ 0 & \frac{\partial \boldsymbol{\theta}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\varphi}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} & \frac{\partial \boldsymbol{\theta}_{\boldsymbol{i,k}}^{\boldsymbol{C_k}}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\theta}_{\boldsymbol{i,k}}^{\boldsymbol{C_{k-1}}}} \end{bmatrix} \tag{C.21}$$

in which $\frac{\partial \varphi_{i,k}^{C_k}}{\partial M}$ is the same as Equation (B.9). $\frac{\partial \theta_{i,k}^{C_k}}{\partial M}$ is the same as Equation (B.10). $\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}}$ and $\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}}$ are given below.

$$\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}} = \boldsymbol{Q^{-1}}(\boldsymbol{r_k^{C_{k-1}}}) \begin{bmatrix} -\cos(\theta_{i,k}^{C_{k-1}})\sin(\varphi_{i,k}^{C_{k-1}}) \\ -\sin(\varphi_{i,k}^{C_{k-1}})\sin(\theta_{i,k}^{C_{k-1}}) \\ \cos(\varphi_{i,k}^{C_{k-1}}) \end{bmatrix} \tag{C.22}$$

$$\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}} = \boldsymbol{Q^{-1}}(\boldsymbol{r_k^{C_{k-1}}}) \begin{bmatrix} -\cos(\varphi_{i,k}^{C_{k-1}})\sin(\theta_{i,k}^{C_{k-1}}) \\ \cos(\varphi_{i,k}^{C_{k-1}})\cos(\theta_{i,k}^{C_{k-1}}) \\ 0 \end{bmatrix} \tag{C.23}$$