Obstacle Detection for Low Flying UAS UsingMonocular Camera

Submitted by

Fan Zhang, B. Eng

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Type in program name as per Graduate Calendar – not department name

Carleton University

Ottawa, Ontario

Abstract

This is the Abstract page.

[Each thesis must contain an abstract. The abstract should be a synopsis providing the essential topics and conclusions of the thesis. The abstract should be inserted immediately before any acknowledgments and the table of contents. Abstracts must not exceed 150 words (master’s) and 350 words (doctoral).]

Acknowledgements

This is the Acknowledgements page.

Table of Contents

Abstract .... **iii**

Acknowledgements .... **iv**

Table of Contents .... **v**

List of Tables .... **vii**

List of Figures .... **viii**

List of Appendices .... **ix**

1 Chapter: INTRODUCTION **1**

1.1 Problem Statement 2

1.2 Objective .... **Error! Bookmark not defined.**

1.3 Thesis Contributions 2

1.4 Thesis Outline .... **Error! Bookmark not defined.**

2 Chapter: BACKGROUND REVIEW ON COMPUTER VISION TECHNOLOGY FOR OBSTACLE DETECTION **4**

2.1 Design requirement overview for low flying UAV .... **Error! Bookmark not defined.**

2.2 Sensors overview for obstacle detection 4

2.2.1 Overview 4

2.2.2 Monocular vision for obstacle detection 4

2.2.3 Limitations of Optical Sensor in Recursive Algorithm 5

2.2.4 GPS and IMU 5

2.3 SLAM as A Sensor Fusion Framework 5

2.3.1 Recursive Probabilistic Estimation using Extended Kalman Filter 6

2.3.2 Properties of SLAM 6

2.3.3 Linearization Error and Consistency 6

2.3.4 Camera Centric Coordinate System 7

2.4 SLAM for Large Scale Maps (Optional) 7

3 Chapter: Algorithm Description **7**

3.1 Camera Centric Inverse Depth Parameterization 7

3.2 Modeling the System with Extended Kalman Filter 8

3.2.1 Full State Vector 8

3.2.2 Prediction (Will be updated to a more complete form) 9

3.2.3 Measurement Model 9

3.2.4 Composition Step (will include more detail formulas) 11

3.3 Adding and Deleting Features 17
3.4 Initialization 12
3.5 Bundle Correction with GPS (not yet implemented) 18
4 Chapter: Experiments with Real Data 18
4.1 Camera Calibration 22
4.2 Equipment Setup and Data Collection 19
4.3 Ground Truth Data Collection and Comparison 22
5 Chapter: Result Discussion 23
5.1 Convergence 23
5.2 Consistency 30
5.3 Reliability 34
5.4 Impact of the Outliers 38
5.5 Use of IMU Improve reliability in harsh flight 40
5.6 How many frames per process is the best? .... **Error! Bookmark not defined.**
6 Chapter: Future work 42
Appendices 68
Appendix A Linearization of EKF 68
A.1 Equation for Initialization 68
A.2 Equations for Measurement Model 68
A.3 Equations for Composition Step 68
Appendix B 70
B.1 Sub-Appendix 70
Bibliography 71

[The Table of Contents is auto-generated based on the use of heading styles 1-8]

List of Tables

This is the List of Tables.

[This list must be included if there are tables in your thesis]

Table 1 Sample caption for Table in Chapter 1 ....**Error! Bookmark not defined.**

Table 2 Sample caption for Table in Chapter 2 ....**Error! Bookmark not defined.**

[This list is auto-generated when captions for tables are created using the 'Insert caption' feature on the References ribbon]

List of Figures

This is the List of Illustrations.

[This list must be included if there are illustrations in your thesis]

Illustration 1 Sample of caption for Illustration in Chapter 1........ **Error! Bookmark not defined.**

Illustration 2 Sample of caption for Illustration in Chapter 2........ **Error! Bookmark not defined.**

[This list is auto-generated when captions for illustrations are created using the 'Insert caption' feature on the References ribbon]

[As you edit your document and add headings, you'll periodically need to update the table of contents, list of tables, and list of illustrations]

List of Appendices

This page lists all of the appendices.

Appendix A……………………. ……..…………………………………………………3
A.1 Sub-Appendix……………………………………………………………….3
A.2 Sub-Appendix……………………………………………………………….3
Appendix B. ………………….....……….…………………………………………………4
B.1 Sub-Appendix……………………………………………………………….4

[This list must be manually entered]

lowest significant bit (LSB)
FOI (field of view)
simulated unmanned aircraft system (SUAS)

1 Chapter: INTRODUCTION

Generating an accurate and high precision model of its surrounding environment to indicate hazard features is an important issue for any autonomous vehicle. Knowing its own location in the map is essentialfor the vehicle to navigate and avoidobstacle autonomously.

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

Traditional 2D SLAM algorithms are well established in the past decade [ref]. A SLAM algorithm typically utilises measurements from several types of sensor which can be divided into two groups, those that provide vehicles pose and orientation measurement, such as wheel odometry, GPS, or IMU; and those that provide landmark bearing and range, measurement, such as radar, sonar, laser range finder. In recent years, optical sensors are actively being incorporated into SLAM algorithm and successfully used in ground vehicle navigation [ref]. For aerial vehicles[1][2][3], the experiments are mostly limited to simulation, and results with realistic aerial video data are unavailable.

1.1 Problem Statement

Obstacle detection has received a lot of research interest in recent years. Various algorithms were developed for ground, under water and aerial vehicle using different sensor such as sonar, radar, LIDAR, and vision. Most research focus on utilizing with only one sensor. Yet, research shows that using multiple sensors produces a better measurement than single sensor [reference in proposal]. With various sensors readily available on most UAV navigation hardware; such as accelerometers, gyroscope, GPS receiver, altimeter, etc., fully utilizing these sensors to aid the main OD sensor helps to improve the accuracy and robustness of the range measurement, especially in harsh flying condition.

This thesis focuses on developing anobstacle detection system by using a SLAM algorithm as a sensor fusion framework for medium size UAV conduction low altitude terrain following flight in natural environment.The obstacles are static objects on ground, and moving objects are not considered. Research presented in this thesis contributes to the project of developing a mid size UAV to perform geological survey, carried out by Carleton in collaborated with Sander Geophysis Ltd., an Ottawa based company specialized in high precision geophysical survey. Toachieve high resolution data acquisition, the UAV must be able to perform terrain following flight with altitude from ground as low as 10 meters at speed ranging from 60 knots (30 m/s) to 100 knots (51.4 m/s). The rate of climb for the UAV is specified to 400ft of vertical rise per minutes (122 meters per minutes). A quick analysis on the UAV specification and aerodynamic behavior reveals the requirement of a practical obstacle detection system. Assuming a tree height of 20 meters, which is the average height for oak or pine, to allow for enough time to avoid obstacle, the UAV must be able to detect the threat at least 610 meters away from it (Figure 1). This analysis indicates that the obstacle detection must be able to map object up to a few thousands away from the UAV.

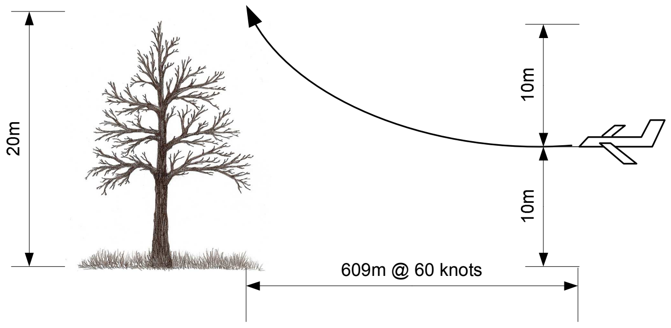


Figure 1 Case study for obstacle detection requirement

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

1.2 Contributions

The thesis reviews the properties and consistency of a typical Extended Kalman Filter (EKF) based SLAM algorithm, and discusses the advantage and limitation of vision based SLAM method. The analysis motivates the development of an improved method by fusing multiple sensors into a mono vision EKF based SLAM framework.

Using a monocular vision for mapping is a bearing only SLAM problem. The measurement is through projection, which loses information about the relative position of the feature since the range is unknown. Without camera motion measurements, map created by monocular vision can be scaled arbitrarily. For a SLAM application in aerial scenario, camera vibration and sudden movement is common when aircraft is hit by cross wind, which can cause the lost of tracked features. A recursive EKF based SLAM algorithm is described in this thesis. The method utilizes sensor onboard the UAV to provide motion measurement of the camera, and improve the robustness of the algorithm under rough flying condition. Real aerial data were collected to test the performance and accuracy of the algorithm in a scenario similar to the one where the UAV will be eventually deployed. The preliminary result of the test flight was published in []. This paper is one of the first ones in the field that successfully applying monocular vision SLAM in large scale aerial application. A more thorough analysis on the behavior of the algorithm and its error is presented in this thesis.Furthermore, a number of baseline separations for the cameraare tested to optimize the performance and computation cost of the algorithm.

1.3 Organization

The thesis is organized as follows:

- Chapter 2 presents an overview on sensors, computer vision algorithms, SLAM algorithm related to obstacle detection and range measurement.
- Chapter 3 describes the detail implementation of the proposed multisensory monocular SLAM algorithm.
- Chapter 4 describes camera calibration procedure, equipments setup for the aerial data collection, and data preparation steps.
- Chapter 5 presents detail analysis on the performance of the algorithm. Convergence and consistency of the algorithm is discussed in section 5.1and 0. Error analysis compared to ground truth data is presented in 0 and 0. The effect of using multiple sensors in improving the robustness is discussed in 5.5. At last, the camera baseline optimization is given in **Error! Reference source not found.**.

2 Chapter: REVIEW ON SENSORS AND RELATED WORK

2.1 SENSORS FOR OBSTACLE DETECTION

2.1.1 Overview

Many work in obstacle detection uses range sensors such as radar, laser range finder, LIDAR, sonar. [reference goes here]. Radar and laser range finder provides only point measurement at a given position and orientation. To acquire a full 3D range map of a scene, mechanical scanning mechanisms are required, which limits the data acquisition rate of these device. LIDAR operate in the same manner as laser range finder, except with the scanning mechanism built in. These sensors usually have high power requirement and mass, and may not be suitable for small and mid size UAV. Sonar is usually used in indoor or under water applications, and have wide beam profile which make it difficult to identify the origin of return signal, and results in low resolution range map. 3D flash LIDAR is capable of acquire 3D range measurement simultaneously by illuminating the entire field of view of the camera with a single laser, and capturing the reflected laser with a 2D imaging sensor [reference; wikipedia]. However, its high cost has limited its use in commercial application.

In recent years, many researches use optical sensor as a passive range sensor for its low weight, low cost. With the help of computer vision technology, optical sensors have been successfully used for range mapping and obstacle detection in a number of platforms [references]. There are several types of configuration in using optical sensor for range mapping: monocular, binocular, or multi-camera. Since optical sensors are bearing only sensors, the principle of range measurement is through triangulation a common scene point in two or more images captured. For binocular camera setups, two cameras are placed apart from each other with their relative geometry known and captures images simultaneously. If the position of a scene point can be accurately found in the images by both cameras, its distance can be calculated by using the difference in position of the projected point in images, and the separation of the cameras.

- Radar, sonar, laser range finder, 3D flash lidarvs. optical sensors
  - Radar, laser range finder have high power requirement and mass
  - .
  - Depth measurement can be obtained through optical sensors, which are inexpensive and light weight
  - Depth maps of a 3-D scene can be computed from a single pair of stereo camera. Stereo processing can require significant computational effort
- Monocular camera characteristic:
  - bearing-only sensor, which provide the measurement on the direction of the feature, and not the range. Other sensors, such as radar, are range and bearing sensors.

2.1.2 Monocular Vision and Binocular Vision

- Optical flow vs. feature detection and tracking
- The correspondence problem
- Initialization problem (addressed by Inverse depth parameterization)
- Lack of scale information of overall map -> must work with other sensors which provide robot motion measurement.

2.1.3 Limitations of Optical Sensor in Recursive Algorithm

- Error Accumulation over Iterations
- Feature quality Decreases over Iterations

2.1.4 GPS and IMU

GPS and IMU are generally available on UAVs. These sensors provide a measurement on the robot motion. Odometry can provide the scale information which is missing in the bearing only measurement. Furthermore, odometry provides some prior information on the robot motion which can help to disambiguate the solution.

2.2 SLAM as A Sensor Fusion Framework

An essential aspect of autonomy for a mobile robot is the capability to determine its location. This capability is known as localization. Localization is typically a prerequisite for accomplishing real tasks, whether it is exploration, navigation toward a known goal, transportation of material, construction or site preparation. In many applications, the mobile robot has an a priori map. Given a map, the robot may localize by matching current sensor observations to features in the map. Given enough features and an unambiguous geometry, the pose of the robot can be determined or at least narrowed down to a set of possible locations.

Usable maps do not always exist, and it is not always possible to have accurate externally referenced pose estimates. If an a priori map is not available, the robot may need to construct one. With a precise, externally referenced position estimate from GPS or similar means, the robot can take its sensor observations, reference the observations to its current pose, and insert features in the map in the appropriate places. Without maps or externally referenced pose information, the robot must produce its own map and concurrently localize within that map. This problem has been referred to as concurrent localization and mapping (CLM) and simultaneous localization and mapping (SLAM).

2.2.1 Recursive Probabilistic Estimation using Extended Kalman Filter

Present the typical EKF model for SLAM problem.

2.2.2 Properties of SLAM

Needs editing.Directly from papers.

Dissanayake[1] proved three important convergency properties of the EKF solution to SLAM, namely that: (1) the determinant of any submatrix 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 and (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.

The properties imply:

- The entire structure of the SLAM problem critically depends on maintaining complete knowledge of the cross correlation between landmark estimates. Minimizing or ignoring cross correlations is precisely contrary to thestructure of the problem. (Early EKF for OD work eliminate the cross correlations between features and veichel pose in an attempt to reduce computation complexity.)
- As the vehicle progresses through the environment the errors in the estimates of any pair of landmarks become more and more correlated, and indeed never become less correlated.
- In the limit, the errors in the estimates of any pair of landmarks become fully correlated. This means that given the exact location of any one landmark, the location of any other landmark in the map can also be determined withabsolute certainty.
- As the map converges in the above manner, the error in the absolute location of every landmark (and thus the whole map) reaches a lower bound determined only by the error that existed when the first observation was made (Initialize the parameters using 1st frame as coordinate origin with minimum variance – Algorithm initialization).[1]

It is important to note that these theoretical results only refer to the evolution of the covariance matrices computed by EKF in the ideal linear case. They overlook the fact that given that SLAM is a nonlinear problem, there is no guarantee that the computed covariance will match the actual estimation erros which is the true SLAM consistency issue. [2]

2.2.3 Linearization Error and Consistency

Many research report filter divergence due to linearization error. Literature review here:

Huang [3] investigate further on properties and consistency of nonlinear two-dimentional EKF based SLAM problem, and conclude:

- Most of the convergence properties in [3] 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 due to (i) the violation of some fundamental constraints governing the relationship between various Jacobians when they are evaluated at the current state estimate, and (ii) the use of relative location information from robot to landmarks to update the absolute robot and landmark location estimates.

The robot orientation uncertainty plays an important role in both the EKF SLAM convergence and the possible inconsistency. In the limit, the inconsistency of EKF SLAM may cause the variance of the robot orientation estimate to be incorrectly reduced to zero.

Linearization error can be interpreted as error resulted from calculating the Jacobian at the estimated state (wrong state) instead of the true state.

2.2.4 Camera Centric Coordinate System

2.3 SLAM for Large Scale Maps (Optional)

3 Chapter: Algorithm Description

This section describes the algorithm. Same as my paper

The algorithm described in Chapter3 Chapter:was implemented in Python programming language. An open source machine vision library OpenCV[8] was utilized to perform feature extraction and tracking. The feature extraction method used was the Shi-Tomasi corner detector[9]. Feature tracking was accomplished through the pyramid implementation of Lucas-Kanade optical flow method [10].

3.1 Camera Centric Inverse Depth Parameterization

The standard way of describing a feature's position is to use the Euclidean XYZ parameterization. In practical outdoor range estimation problem, the algorithm must deal with features located at near infinity. Inverse depth parameterization overcomes this problem by representing range d in its inverse form $\rho = \frac{1}{d}$. In addition, features at infinity contribute in estimating the camera rotational motion even though they offer little information on camera translational motion. Furthermore, inverse depth parameterization allows us to initialize features into the EKF framework before it is safely triangulated.

The inverse depth parameterization used in this work was first introduced in [4]. All features and camera positions are referred to a world reference frame. When used in an Extended Kalman Filter framework, the system suffers decreasing linearity when camera is moving away from the world origin. A modified method which uses the camera center as the origin was proposed in [5]. Our work has adopted the camera centered approach with minor modification to integrate the inertial measurements.

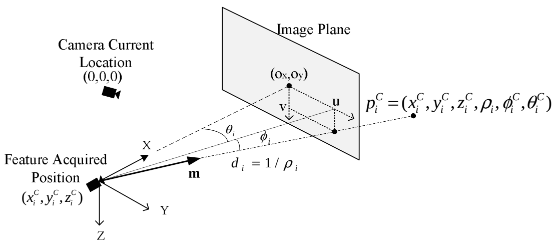


Image Plane
Camera Current Location (0,0,0)
(o_x,o_y)
u
v
$p_i^C = (x_i^C, y_i^C, z_i^C, \rho_i, \phi_i^C, \theta_i^C)$
θ_i
ϕ_i
X
Feature Acquired Position (x_i^C, y_i^C, z_i^C)
m
$d_i = 1/\rho_i$
Y
Z

Figure 2 Inverse Parameterization

A scene point p_i^C can be defined by 6 parameters, with the superscript C representing a camera reference frame. (Figure 2):

$$p_i^C = [x_i^C \quad y_i^C \quad z_i^C \quad \rho_i \quad \phi_i^C \quad \theta_i^C]$$

The first three parameters $[x_i^C, y_i^C, z_i^C]$ represent the initial position where the feature is first observed. ρ_i is the inverse distance from the initialization position to the feature. The elevation-azimuth pair $[\phi_i^C, \theta_i^C]$ encodes a unit vector pointing from the initialization point to the feature. The vector is given by

$$m(\phi_i^C, \theta_i^C) = \begin{bmatrix} \cos\phi_i^C \cos\theta_i^C \\ \cos\phi_i^C \sin\theta_i^C \\ \sin\phi_i^C \end{bmatrix}$$

3.2 Modeling the System with Extended Kalman Filter

3.2.1 Full State Vector

The EKF state vector is defined as

$$x = [OX_W^C \quad c^C \quad r^C \quad p_1^C \quad p_2^C \quad \ldots]^T$$

Equation 1

where $OX_W^C = [O_x^C \quad O_y^C \quad O_z^C W_x^C \quad W_y^C \quad W_z^C]^T$ contains translation parameters $O_{x,y,z}^C$ and rotation parameters $W_{x,y,z}^C$ to transform the camera reference frame to the world reference frame. $[c^C, r^C]^T$ represents the camera translation and rotation motion frame by frame in Euclidean coordinates, and p_i^C contains the feature parameters as described in the previous section.

3.2.2 Prediction (Will be updated to a more complete form)

For a prediction step at time k , the world frame and features parameters are kept unchanged from time $k-1$. The camera parameters are updated using the new inertial measurements: velocity v^C , acceleration a^C , and rate of change in roll/pitch/yaw w^C . The camera motion parameters at time k are then

Equation 2

$$\begin{bmatrix} OX_W^C \\ c^C \\ r^C \\ p_1^C \\ p_2^C \\ \vdots \end{bmatrix}_{k+1} = \begin{bmatrix} OX_W^C \\ c^C_{measured} \\ r^C_{measured} \\ p_1^C \\ p_2^C \\ \vdots \end{bmatrix} \quad \textbf{or} \quad \begin{bmatrix} OX_W^C \\ c^C \\ r^C \\ p_1^C \\ p_2^C \\ \vdots \end{bmatrix}_{k+1} = \begin{bmatrix} OX_W^C \\ c^C \\ r^C \\ p_1^C \\ p_2^C \\ \vdots \end{bmatrix}_{k}$$

Where

$c_k^C = v^C \Delta t + \frac{1}{2} a^C \Delta t^2$ $r_k^C = r_{k-1}^C + w^C$

Writing the above in the form of $x_k = \mathrm{F}_{k-1} x_{k-1} + B_k u_k + w_{k-1}$, without INS measurement, F_k is simply I . With INS input,

x↓k = [■(■(I↓(6 × 6)&&@&0↓(3 × 3)&@&&0↓(3 × 3))&0@0&■(I↓(6 × 6)&&@& ⋱ &@&&I↓(6 × 6)))] [■(■([illegible]

3.2.3 Measurement Model

Each observed feature is related to the camera motion through the measurement model (Figure 3). This relationship enables a correction on the camera motion and features parameters based on the features' locations observed in the image.

For a feature p_i^C , the vector h^R pointing from the predicted camera location to the feature initialization position is

$h_k^R = \begin{bmatrix} x_i^C \\ y_i^C \\ z_i^C \end{bmatrix}_k - \begin{bmatrix} c_x^C \\ c_y^C \\ c_z^C \end{bmatrix}_k$	**Equation 3**

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

$h_k^C = Q^{-1}\left(r_k^C\right)\left(\rho_k h_k^R + m\left(\phi_k^C, \theta_k^C\right)\right)$	**Equation 4**

where $Q^{-1}\left(\boldsymbol{r}_k^C\right)$ 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 feature location on image plane can be found by

$$\boldsymbol{h}_k^U = \begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} \dfrac{s_x \boldsymbol{h}_{y,k}^C}{\boldsymbol{h}_{x,k}^C} \\ \dfrac{s_y \boldsymbol{h}_{z,k}^C}{\boldsymbol{h}_{x,k}^C} \end{bmatrix}$$

Equation 5

where s_x and s_y is the scaling factor of the projection, obtained through camera calibration.

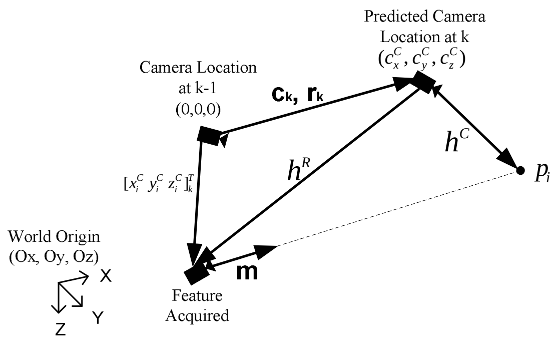


Figure 3 Measurement Model

3.2.4 Composition Step (will include more detail formulas)

Update step corrects the camera motion and feature location in camera frame k-1. To continue to the next cycle of tracking, all parameter must be transform to camera frame k. World reference point coordinate and orientation from k-1 to k is related by

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

Equation 6

$$\begin{bmatrix} W_x^{C_k} \\ W_y^{C_k} \\ W_z^{C_k} \end{bmatrix}_{k-1} = \begin{bmatrix} W_x^{C_{k-1}} \\ W_y^{C_{k-1}} \\ W_z^{C_{k-1}} \end{bmatrix}_k - r^{C_{k-1}}$$

Equation 7

Feature parameters in new camera frame are related to the previous frame by

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

Equation 8

$$[\blacksquare(\rho_\downarrow i @ \phi_\downarrow i^\uparrow(C_\downarrow k)@\theta_\downarrow i^\uparrow(C_\downarrow k))]_\downarrow k = (\blacksquare(\rho_\downarrow(i,k)@m^\uparrow(-1)\,(R^\uparrow(-1)\,(r^\uparrow(C_\downarrow(k-1)))\,)m(\phi_\downarrow(i,k)^\uparrow(C_\downarrow(k-1)\,),\theta_\downarrow(i,k$$

Equation 9

where $m\left(\phi_{i,k}^{C_{k-1}},\theta_{i,k}^{C_{k-1}}\right)$ is the unit vector pointing from the initialization point to the feature seen by the camera at step k-1

The covariance matrix is also affected by this transform. Therefore must be updated. The new covariance matrix is related to the old one by

$$P_k^{C_k} = J_{C_{k-1}\to C_k} P_k^{C_{k-1}} J_{C_{k-1}\to C_k}^T$$

Equation 10

The calculation of $J_{C_{k-1}\to C_k}$ is the same as the linearization of prediction matrix in section **Error! Reference source not found.** Method 2.

In order to apply the correction and update the camera reference frame to the new camera position, an additional composition step is necessary. The world reference frame parameters and features parameters are updated by applying reference frame transformation from the camera location at time k-1 to camera location at time k. The EKF covariance matrix P_k is also updated through

$$P_k = JP_kJ^T \qquad (8)$$

where J is the Jacobian of the composition equations.

3.3 Initialization

3.3.1 Initialize the State Vector

State vectors are initialized at the first frame. The world origin coordinate and orientation, camera motions, and the feature reference points are all initialized to zeros, with variance equals to the smallest machine number. Thus,

$$OX_W^C = [0\ 0\ 0\ 0\ 0\ 0]$$

Equation 11

$$c^C = [0\ 0\ 0]$$

Equation 12

$$r^C = [0\ 0\ 0]$$

Equation 13

$$p_i^C = [0\ 0\ 0\ \rho_i \phi_i \theta_i]$$

Equation 14

The inverse distance ρ of all features are initialized to 0.1 because we are dealing with long distance object. The features elevation-azimuth pair $[\phi_i^C, \theta_i^C]$ is extracted from features coordinates in image plane. First, a vector pointing from camera optical center to feature can be defined by

$$h^C = \begin{bmatrix} h_x^C \\ h_y^C \\ h_z^C \end{bmatrix} = \begin{bmatrix} 1 \\ u \cdot s_x \\ v \cdot s_y \end{bmatrix}$$

Equation 15

Where $[u\ v]$ is the feature coordinate in the image, and $[s_x s_y]$ is the scaling factor of the projection from the scene to image plane. The elevation-azimuth pair $[\phi_i^C, \theta_i^C]$ can then be directly calculated from h^C

$$\phi = \arctan\left(\frac{h_z^C}{\sqrt{{h_x^C}^2 + {h_y^C}^2}}\right)$$

Equation 16

$$\theta = \arctan\left(\frac{h_y^C}{h_x^C}\right)$$

Equation 17

3.3.2 Initialized the State Covariance Matrix

Because the world origin is defined at the first frame, it enables initializing the filter with minimum variance, which helps reducing the lower bound of the filter error. The covariance matrix of the world coordinate and orientation, and the camera motion is

$$P = I_{12\times12} \cdot \varepsilon$$

Equation 18

where I is a 12-by-12 identity matrix, and ε is the lowest significant bit (LSB) of a machine.

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

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

Equation 19

where P_{old} isthe covariance matrix of the existing state vector, and the initial P_{old} is defined in Equation 18. Matrix R is the covariance matrix of the variable in features initialization Equation 14 - Equation 17.

$$R = \begin{bmatrix} \sigma_{x_i^C} & & & & & \\ & \sigma_{y_i^C} & & & 0 & \\ & & \sigma_{z_i^C} & & & \\ & & & \sigma_\rho & & \\ & 0 & & & \sigma_{image} & \\ & & & & & \sigma_{image} \end{bmatrix} = \begin{bmatrix} \varepsilon & & & & & \\ & \varepsilon & & & 0 & \\ & & \varepsilon & & & \\ & & & 0.1 & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{bmatrix}$$

Equation 20

where $[\sigma_{\downarrow}(x_{\downarrow}i^{\uparrow}C\ )\ \sigma_{\downarrow}(y_{\downarrow}i^{\uparrow}C\ )\ \sigma_{\downarrow}(z_{\downarrow}i^{\uparrow}C\ )\ ]$ is the uncertainty of the camera optical center position, initialized to ε . σ_{image} is the image plane pixel variance, set to 1. σ_ρ is the uncertainty of the inverse distance. Because the filter mainly deals with distance feature, σ_ρ is initialized to 0.1 to cover any distance from 50 meters to infinity.

J inEquation 19 is the Jacobian matrix for the initialization equation.

$$J = \begin{bmatrix} & & I & & & & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \frac{\partial p_i}{\partial OX_W^C} & \frac{\partial p_i}{\partial c^C} & \frac{\partial p_i}{\partial r^C} & 0 & \dots & 0 & \frac{\partial p_i}{\partial g_i} \end{bmatrix}$$

Equation 21

Whenever a new feature is added, it's initial position is always $[0\ 0\ 0]$ in the camera centric coordinate, and the $[\rho\ \ \phi\ \ \theta\ ]$ parameters are not a function of OX_W^C , c^C ,or r^C , therefore

$$\frac{\partial p_i}{\partial OX_W^C} = 0_{6\text{x}6}$$

Equation 22

$$\frac{\partial p_i}{\partial c^C} = 0_{6\text{x}3}\ \frac{\partial p_i}{\partial r^C} = 0_{6\text{x}3}$$

Equation 23

J canthen be simplified as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \dfrac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{g}_i} \end{bmatrix}$$

Equation 24

Where $\boldsymbol{g}_i$ includes the variable in matrix $\boldsymbol{R}$: $\boldsymbol{g}_i = [x_i^C \quad y_i^C \quad z_i^C \quad \rho_i \quad u_i \quad v_i]$. Then

$$\frac{\partial \rho_i}{\partial \rho_i} \& \frac{\partial \rho_i}{\partial u_i} \& \frac{\partial \rho_i}{\partial v_i} @ \frac{\partial \phi_i^C}{\partial \rho_i} \& \frac{\partial \phi_i^C}{\partial u_i} \& \frac{\partial \phi_i^C}{\partial v_i} @ \frac{\partial \theta_i^C}{\partial \rho_i} \& \frac{\partial \theta_i^C}{\partial u_i} \& \frac{\partial \theta_i^C}{\square}$$

Equation 25

Based on the rule of derivation,

$$\frac{\partial \phi_i^C}{\partial u_i} = \frac{\partial \phi_i^C}{\partial \boldsymbol{h}^C} \frac{\partial \boldsymbol{h}^C}{\partial u_i}$$

$$\frac{\partial \theta_i^C}{\partial u_i} = \frac{\partial \theta_i^C}{\partial \boldsymbol{h}^C} \frac{\partial \boldsymbol{h}^C}{\partial u_i}$$

$$\frac{\partial \phi_i^C}{\partial v_i} = \frac{\partial \phi_i^C}{\partial \boldsymbol{h}^C} \frac{\partial \boldsymbol{h}^C}{\partial v_i}$$

$$\frac{\partial \theta_i^C}{\partial v_i} = \frac{\partial \theta_i^C}{\partial \boldsymbol{h}^C} \frac{\partial \boldsymbol{h}^C}{\partial v_i}$$

Equation 26

and

$$(\boldsymbol{h})_x^{C^2} + h_y^{C^2} + h_z^{C^{\square}} \uparrow$$

Equation 27

$$\frac{\partial \theta_i^C}{\partial \boldsymbol{h}^C} = \begin{bmatrix} \dfrac{\partial \theta_i^C}{\partial \boldsymbol{h}_x^C} & \dfrac{\partial \theta_i^C}{\partial \boldsymbol{h}_y^C} & \dfrac{\partial \theta_i^C}{\partial \boldsymbol{h}_z^C} \end{bmatrix} = \begin{bmatrix} -\dfrac{\boldsymbol{h}_y^C}{\boldsymbol{h}_x^{C^2} + \boldsymbol{h}_y^{C^2}} & \dfrac{\boldsymbol{h}_x^C}{\boldsymbol{h}_x^{C^2} + \boldsymbol{h}_y^{C^2}} & \boldsymbol{0} \end{bmatrix}$$

Equation 28

$$\frac{\partial \boldsymbol{h}^C}{\partial u_i} = \begin{bmatrix} \dfrac{\partial \boldsymbol{h}_x^C}{\partial u_i} \\ \dfrac{\partial \boldsymbol{h}_y^C}{\partial u_i} \\ \dfrac{\partial \boldsymbol{h}_z^C}{\partial u_i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ s_x \\ \boldsymbol{0} \end{bmatrix}$$

Equation 29

$$\frac{\partial \boldsymbol{h}^C}{\partial v_i} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_x^C}{\partial v_i} \\ \frac{\partial \boldsymbol{h}_y^C}{\partial v_i} \\ \frac{\partial \boldsymbol{h}_z^C}{\partial v_i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s_y \end{bmatrix}$$

Equation 30

3.4 Linearization of Measurement Model

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

$\partial \boldsymbol{h}_{\vdots 1}$

Equation 31

Elements in the matrix above are given in Equation 32 - Equation 42. The formula for calculating Q^{-1} in Equation 39 - Equation 42 can be found in Appendix B

$$\frac{\partial \boldsymbol{h}_k^U}{\partial \boldsymbol{h}_k^C} = \begin{bmatrix} -\frac{s_x \boldsymbol{h}_{y,k}^C}{{\boldsymbol{h}_{x,k}^C}^2} & \frac{s_x}{\boldsymbol{h}_{x,k}^C} & 0 \\ -\frac{s_y \boldsymbol{h}_{z,k}^C}{{\boldsymbol{h}_{x,k}^C}^2} & 0 & \frac{s_y}{\boldsymbol{h}_{x,k}^C} \end{bmatrix}$$

Equation 32

$$\frac{\partial \boldsymbol{h}^C}{\partial \boldsymbol{c}^C} = -R^{-1}(\boldsymbol{r}_k^C)\rho_k$$

Equation 33

$$\frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{r}^C} = \begin{bmatrix} \frac{\partial \boldsymbol{h}^C}{\partial r_x^C} & \frac{\partial \boldsymbol{h}^C}{\partial r_y^C} & \frac{\partial \boldsymbol{h}^C}{\partial r_z^C} \end{bmatrix}$$

Equation 34

$(\partial h^{\uparrow}C)/(\partial r_{\downarrow}x) = [\blacksquare(0@\boldsymbol{h}_{\downarrow}x^{\uparrow}C\ (sin(r_{\downarrow}x)sin(r_{\downarrow}z) + cos(r_{\downarrow}x)cos(r_{\downarrow}z)sin(r_{\downarrow}y)) + \boldsymbol{h}_{\downarrow}y^{\uparrow}C\ (-cos(r_{\downarrow}z)sin(r_{\downarrow}x) + cos(r_{\downarrow}x)sin(r_{\downarrow}y)$

Equation 35

$(\partial h^{\uparrow}C)/(\partial r_{\downarrow}y) = [\blacksquare(-\boldsymbol{h}_{\downarrow}z^{\uparrow}C\ cos(r_{\downarrow}y) - \boldsymbol{h}_{\downarrow}x^{\uparrow}C\ cos(r_{\downarrow}z) * sin(r_{\downarrow}y) - \boldsymbol{h}_{\downarrow}y^{\uparrow}C\ sin(r_{\downarrow}y)sin(r_{\downarrow}z)@ - \boldsymbol{h}_{\downarrow}z^{\uparrow}C\ sin(r_{\downarrow}x)sin(r_{\downarrow}y) +$

Equation 36

r_y

Equation 37

$$\frac{\partial h^C}{\partial p_i} = \begin{bmatrix} \frac{\partial h^C}{\partial x_i^C} & \frac{\partial h^C}{\partial y_i^C} & \frac{\partial h^C}{\partial z_i^C} & \frac{\partial h^C}{\partial \rho_i} & \frac{\partial h^C}{\partial \theta_i} & \frac{\partial h^C}{\partial \phi_i} \end{bmatrix}$$

Equation 38 is a 3x6 matrix

$$\begin{bmatrix} \frac{\partial h^C}{\partial x_i^C} & \frac{\partial h^C}{\partial y_i^C} & \frac{\partial h^C}{\partial z_i^C} \end{bmatrix} = Q^{-1}(r_k^C)\rho_k$$

Equation 39

$$\frac{\partial h^C}{\partial \rho_i} = -Q^{-1}(r_k^C) \cdot c_k$$

Equation 40

$$\frac{\partial h^C}{\partial \theta_i} = Q^{-1}(r_k^C) \begin{bmatrix} -\cos(\phi_i)\sin(\theta_i) \\ \cos(\phi_i)\cos(\theta_i) \\ 0 \end{bmatrix}$$

Equation 41

$$\frac{\partial h^C}{\partial \phi_i} = Q^{-1}(r_k^C) \begin{bmatrix} -\cos(\phi_i)\sin(\theta_i) \\ -\sin(\phi_i)\sin(\theta_i) \\ \cos(\phi_i) \end{bmatrix}$$

Equation 42

3.5 Jacobian Matrix of Composition Equations

$$J_{C_{k-1} \to C_k} = \frac{\partial x_k}{\partial x_{k-1}} = \begin{bmatrix} \frac{\partial x_k}{\partial OX_{W,k-1}^C} & \frac{\partial x_k}{\partial c_{k-1}} & \frac{\partial x_k}{\partial r_{k-1}} & \frac{\partial x_k}{\partial p_{1,k-1}^C} & \frac{\partial x_k}{\partial p_{2,k-1}^C} & \cdots \end{bmatrix}$$

Equation 43

$\frac{\partial x_k}{\partial OX_{W,k-1}^C}$ is given by:

Equation 44

Derivatives of x_k with respect to c and p are given in Equation 45 - Equation 46.

[■((∂O↓(W,k)↑C)/(∂c↓(k − 1))&(∂O↓(W,k)↑C)/(∂r↓(k − 1))@■((∂W↓(W,k)↑C)/(∂c↓(k − 1))@(∂c↓k)/(∂c↓(k −

Equation 45

where

$$pr = \begin{bmatrix} 1 & 0 & 0 \\ & \frac{\partial \phi_{i,k}^C}{\partial r_{k-1}} & \\ & \frac{\partial \theta_{i,k}^C}{\partial r_{k-1}} & \end{bmatrix} = \begin{bmatrix} 1 \quad 0 \quad 0 \\ \frac{\partial m^{-1}\left(Q^{-1}\left(r_k^{C_{k-1}}\right) m\left(\phi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}\right)\right)}{\partial r_{k-1}} \end{bmatrix}$$

Equation 46

Let $M = Q^{-1}\left(r_k^{C_{k-1}}\right) m\left(\phi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}\right)$, and $m^{-1} = m^{-1}(M)$, then

$$\frac{\partial m^{-1}}{\partial r_k^{C_{k-1}}} = \frac{\partial m^{-1}}{\partial M} \cdot \frac{\partial M}{\partial r_k^{C_{k-1}}}$$

Equation 47

where

$$\frac{\partial M}{\partial r_k^{C_{k-1}}} = \frac{\partial Q^{-1}\left(r_k^{C_{k-1}}\right)}{\partial r_{k-1}} \cdot m\left(\phi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}\right)$$

Equation 48

as m^{-1} isthe function that calculate $[\phi\ \theta]$ from a vector, $\frac{\partial m^{-1}}{\partial M}$ is the same as Equation 27 and Equation 28.

Derivatives of x_k with respect to feature parameters p_i^C are given in Equation 49- Equation 53

$C_{\square\downarrow}$

Equation 49

$(\partial p_\downarrow(i,k)^\uparrow(C_\downarrow(k\ ))) / (\partial p_\downarrow(i,k)^\uparrow(C_\downarrow(k-1\ )\ )\ ) = [\blacksquare((\partial x_\downarrow(i,k)^\uparrow(C_\downarrow k\ )) / (\partial p_\downarrow(i,k)^\uparrow(C_\downarrow(k-1)\ )\ ) @ (\partial y_\downarrow(i,k)^\uparrow C) / (\partial p_\downarrow$

Equation 50

$(\partial 〖p(3:5)〗_\downarrow(i,k)^\uparrow(C_\downarrow k\ )) / (\partial 〖p(3:5)〗_\downarrow(i,k)^\uparrow(C_\downarrow(k-1)\ )\ ) = [\blacksquare(1\&0\&0@0\&(\partial\phi_\downarrow(i,k)^\uparrow(C_\downarrow k\ )) / (\partial\phi_\downarrow(i,k)^\uparrow(C_\downarrow($

Equation 51

$\frac{\partial\phi_{i,k}^{C_k}}{\partial M}$ is the same as Equation 27. $\frac{\partial\theta_{i,k}^{C_k}}{\partial M}$ is the same as Equation 28. $\frac{\partial M}{\partial\phi_{i,k}^{C_{k-1}}}$ and $\frac{\partial M}{\partial\theta_{i,k}^{C_{k-1}}}$ is given below.

$$\frac{\partial M}{\partial\phi_{i,k}^{C_{k-1}}} = R^{-1}(r^C)\begin{bmatrix} -\cos\left(\theta_{i,k}^{C_{k-1}}\right)\sin\left(\phi_{i,k}^{C_{k-1}}\right) \\ -\sin\left(\phi_{i,k}^{C_{k-1}}\right)\sin\left(\theta_{i,k}^{C_{k-1}}\right) \\ \cos\left(\phi_{i,k}^{C_{k-1}}\right) \end{bmatrix}$$

Equation 52

$$\frac{\partial M}{\partial\theta_{i,k}^{C_{k-1}}} = R^{-1}(r^C)\begin{bmatrix} -\cos\left(\phi_{i,k}^{C_{k-1}}\right)\sin\left(\theta_{i,k}^{C_{k-1}}\right) \\ \cos\left(\phi_{i,k}^{C_{k-1}}\right)\cos\left(\theta_{i,k}^{C_{k-1}}\right) \\ 0 \end{bmatrix}$$

Equation 53

3.6 Adding and Deleting Features

Features that move out of the camera's FOI are removed from the filter. The removal is done by directly deleting the parameters from the state vector, and the related rows and columns from the state covariance matrix. The parameters of the deleted features are still recorded into the database, and remained unchanged for all iteration after the removal unless a GPS bundle correction occurs.

To maintain sufficient amount of features to generate map, new features are acquired and added to the filter if a feature deletion procedure occurred, or number of tracked feature is lower than a threshold. The feature addition procedure occurs after the composition step of an iteration, and follows the same procedure as described in section 3.3.

3.7 Bundle Correction with GPS (not yet implemented)

Apply overall correction on the map at a sparser time interval using the GPS positions.

3.8 Processing for Comparison to Ground Truth Data

The accuracy of the result was analyzed by comparing to the digital elevation models (DEM) [9].To make the comparison, the features coordinate and the DEM data must be brought to the same coordinate system. For ease of viewing, the world coordinate is used. The features' positions can be converted to Euclidean representation in world frame by:

$$\begin{bmatrix} X_i^W \\ Y_i^W \\ Z_i^W \end{bmatrix} = Q^{-1}\left(O_{XYZ}^c, W_{XYZ}^c\right)\left(\begin{bmatrix} x_i^C \\ y_i^C \\ z_i^C \end{bmatrix} + \frac{1}{\rho_i} m\left(\phi_i^C, \theta_i^C\right)\right)$$

The DEM is converted into the world frame using the UAV's initial GPS location$[Latitude_{init} \quad Longitude_{init} \quad Height_{init}]$ and orientation$[Roll_{init} \quad Pitch_{init} \quad Azimuth_{init}]$. First, the UAV's GPS latitude and longitude is converted into UTM representation$[Northing_{init} \& Easting_{init} \& Height_{init}]$. Then, a transformation matrix can be defined as followed:

$$Q_{Roll}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(Roll_{init}) & \sin(Roll_{init}) & 0 \\ 0 & -\sin(Roll_{init}) & \cos(Roll_{init}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{Pitch}^{-1} = \begin{bmatrix} \cos(Pitch_{init} + \pi) & 0 & -\sin(Pitcch_{init} + \pi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(Pitch_{init} + \pi) & 0 & \cos(Pitch_{init} + \pi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{Azimuth}^{-1} = \begin{bmatrix} \cos\left(Azimuth_{init} + \frac{\pi}{2}\right) & \sin\left(Azimuth_{init} + \frac{\pi}{2}\right) & 0 & 0 \\ -\sin\left(Azimuth_{init} + \frac{\pi}{2}\right) & \cos\left(Azimuth_{init} + \frac{\pi}{2}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -Northing_{init} \\ 0 & 1 & 0 & -Easting_{init} \\ 0 & 0 & 1 & -Height_{init} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q^{-1} = Q_T^{-1} \cdot Q_{Roll}^{-1} \cdot Q_{Pitch}^{-1} \cdot Q_{Azimuth}^{-1}$$

At last, the DEM in world coordinate is given by

$$DEM_{World} = Q^{-1} \cdot DEM_{UTM}$$

0612

4 Chapter: Experiments with Real Data

4.1 Equipment Setup and Data Collection

In order to examine the accuracy and feasibility of the algorithm being used in low flying UAV for obstacle detection purpose, realistic aerial video and navigation data are collected through a test flight with the support of Sander Geophysics Ltd. A main purpose of the test flight is to obtain a piece of aerial video with the camera close to the ground as much as possible. This is difficult to achieve with any manned fixed wing aircraft. Therefore, a simulated unmanned aircraft system (SUAS) was used to carry all sensors. The SUAS is then towed by a helicopter via a tow rope of 33 meters long(

Figure 4)to complete the survey. Yet, to prevent the SUAS from being caught by tree top, sufficient clearance must be left between the SUAS and the vegetation. As a result, the helicopter flew a planned path at approximately 100 meters above ground, and SUAS at approximately 70 meters above ground.

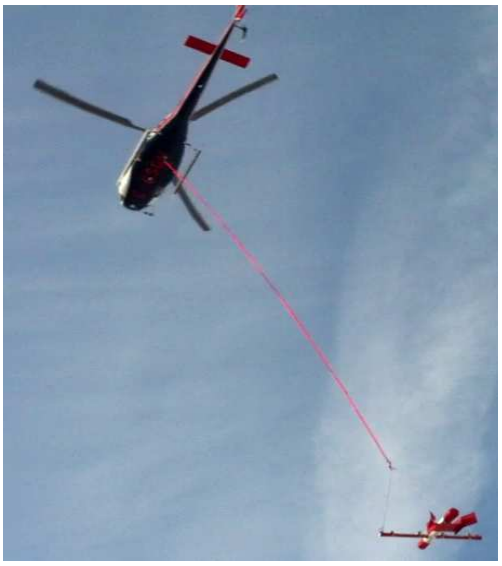


Figure 4 Simulated UAS towed by helicopter

Sensors mounted on the SUAS included one wide angle CCD camera with 6 mm focal length lens capturing monocular image sequence at 30 fps, a pair of narrow angle CCD cameras for binocular images, one GPS antenna, and one INS/GPS navigation unit Athena GS-111m (Figure 5). Analog video and navigation data are sent to the helicopter via three BNC cables and one data cable. Installed in the helicopter are two SGL data acquisition system CDAC. This system records video and data from SUAS, as well as data from sensors installed on the helicopter, including GPS, radar and laser altimeter, air pressure, temperature, humility, etc. Navigation data from the SUAS were sent in RS485, and were directly recorded via the UART port of the computer (Figure 6). Videos from the three cameras were digitized to 720x480 resolution images using a PC/104+ MPEG4 video encoder from Parvus installed in CDAC. The video were time-stamped with GPS second on the image screen for post-flight synchronization with the navigation measurements. A snapshot of the digitized video is shown in Figure 7.

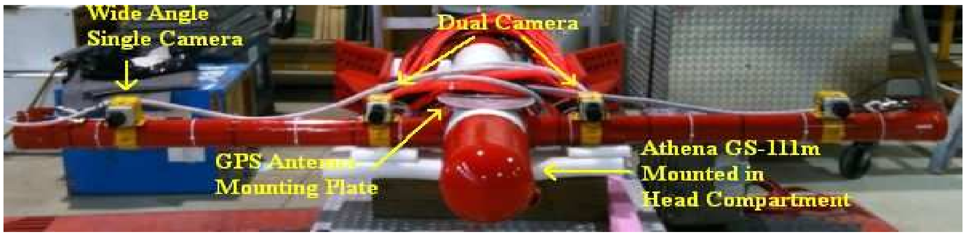

Wide Angle Single Camera
Dual Camera
GPS Antenna Mounting Plate
Athena GS-111m Mounted in Head Compartment

Figure 5 Sensors mounted on Simulated UAS. Top: the SUAS, bottom left: Athena GS-111m, bottom right: GPS antenna

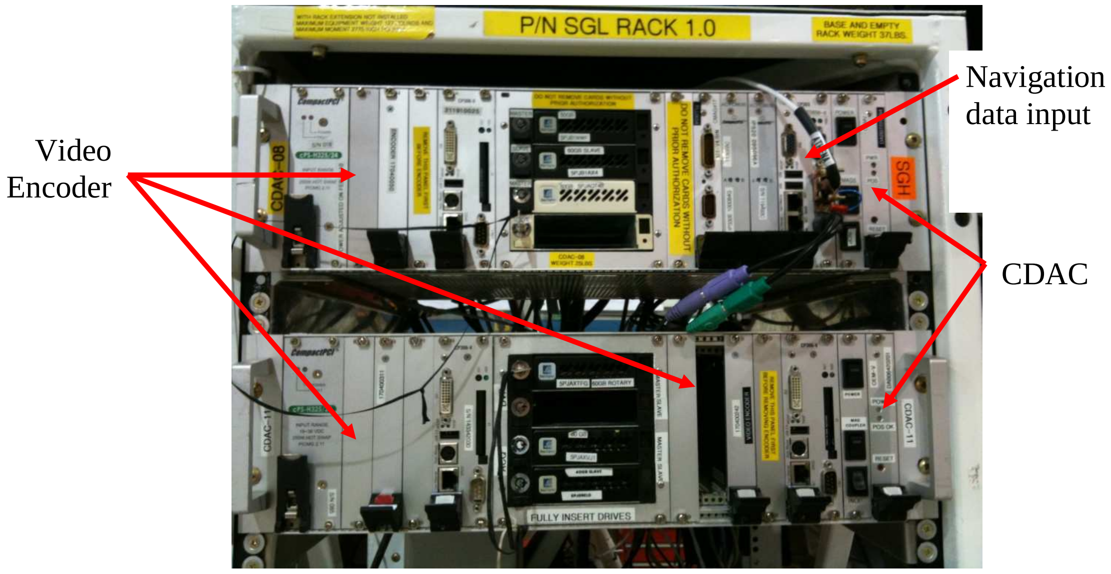


Figure 6 Data Acquisition System

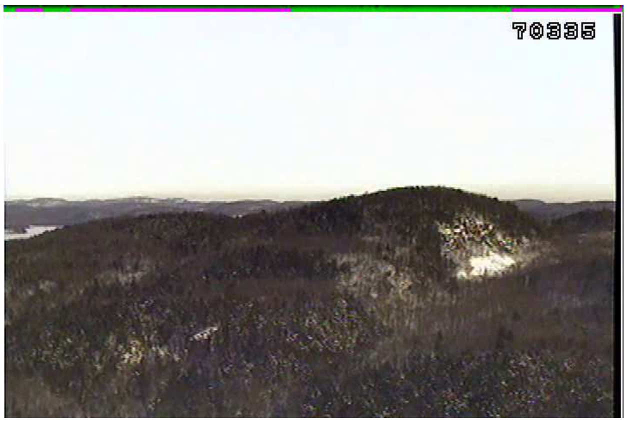


Figure 7 Image from monocular camera with GPS second timestamp

4.2 Camera Calibration

4.3 Ground Truth Data Collection and Comparison

5 Chapter: Result From Flight Data

Since all parameters are tracked in camera frame, their value is different when viewed from a fixed point in world frame. Therefore, all parameters are converted back to world frame before plotting.

5.1 Convergence

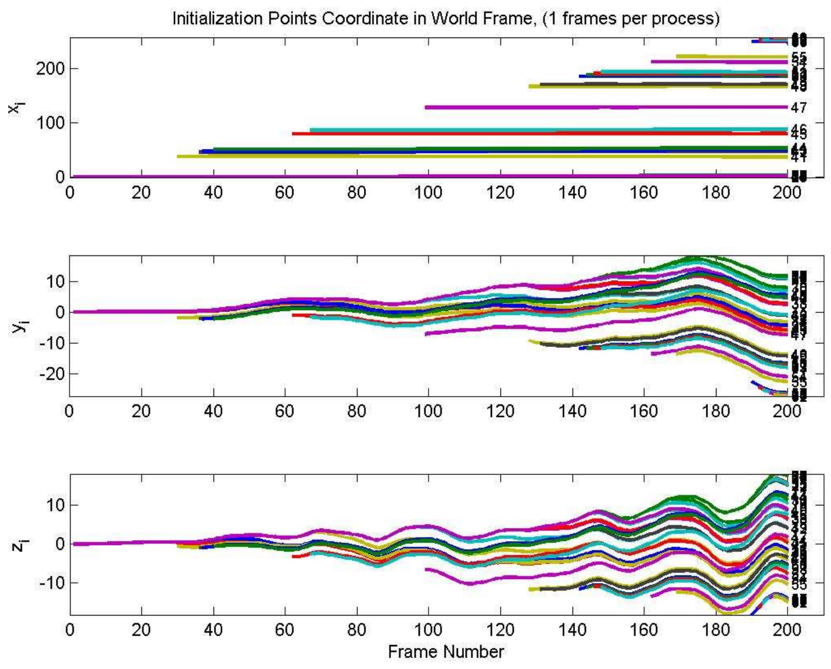


Figure 8 Initialization points coordinates (in world frame) of all features

Figure 8 illustrate the tracking of feature initialization points $[x_i, y_i, z_i]$ in world frame. Ideally, the coordinates should converge to a fixed value. However, the plot indicates a variation as the vehicle travels along. The plots for xi appear to be quite flat because of the large scale of xi value. Figure 9 shows a more detail xi drift by removing the mean value from the recorded sequence. Comparing the pattern of the drift, it is highly correlated to the aircraft pitch and yaw rotation (figure?). The significant of the drift is measured by the maximum drift seen throughout the 200 processed frame, and are listed below (table?). The maximum drift is defined by:

Maxdrift = max(abs(initial value – maximum), abs(initial value – minimum))

Maxmum xi drift	Maximum yi drift	Maximum zi drift

2.21 meters	18.09 meters	18.45 meters

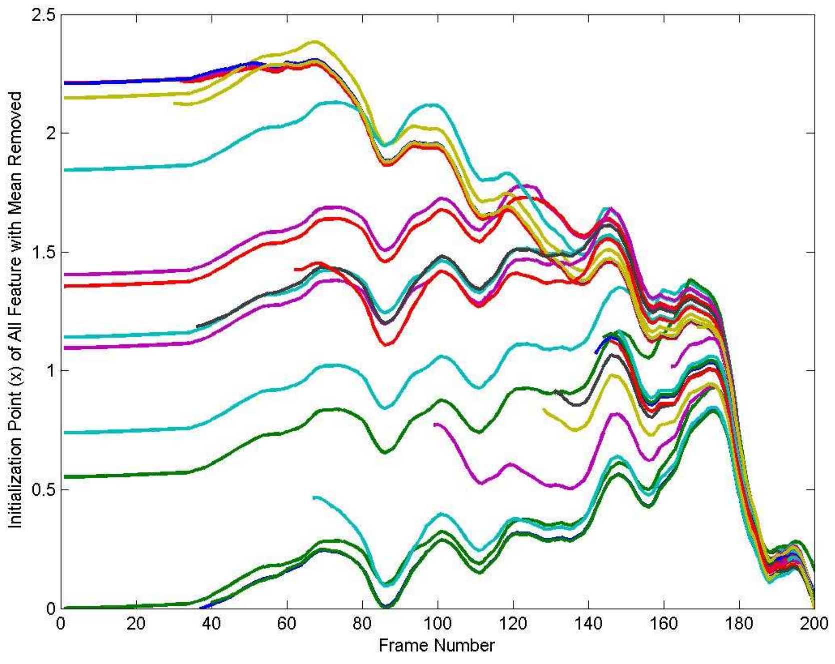


Figure 9 x_i with mean removed

Although xi also drifts, its magnitude is significantly less than y and z. With maximum drift of 2.2 meters, and mean drift of ?meters, considering the magnitude of the initial value for xi, it actually is quite stable.

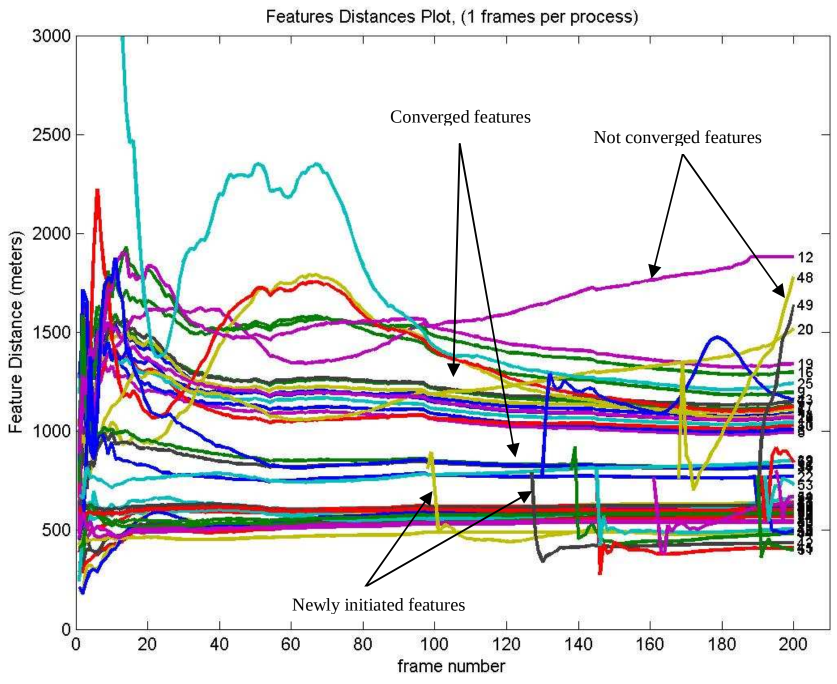


Figure 10 Feature Distance Convergence Plot (1 frame per process)

Figure 10 shows the convergence behavior of inverse depth over the 200 frames processed. The plot includes features initialized at first frame, as well as those added later on. Features that moved out of the field of view of the camera were still kept in the plot, with their value kept unchanged. A total of 60 features were initialized and tracked. Most features converged to a value within the first 50 frames and remained stable. ? features took longer to converge, but started to settle in after 120 frames. These features are those on the edge of mountain (confirm required).Some other features (? Out of 60) never show any convergence behavior. What are these features, and why they don’t converge?

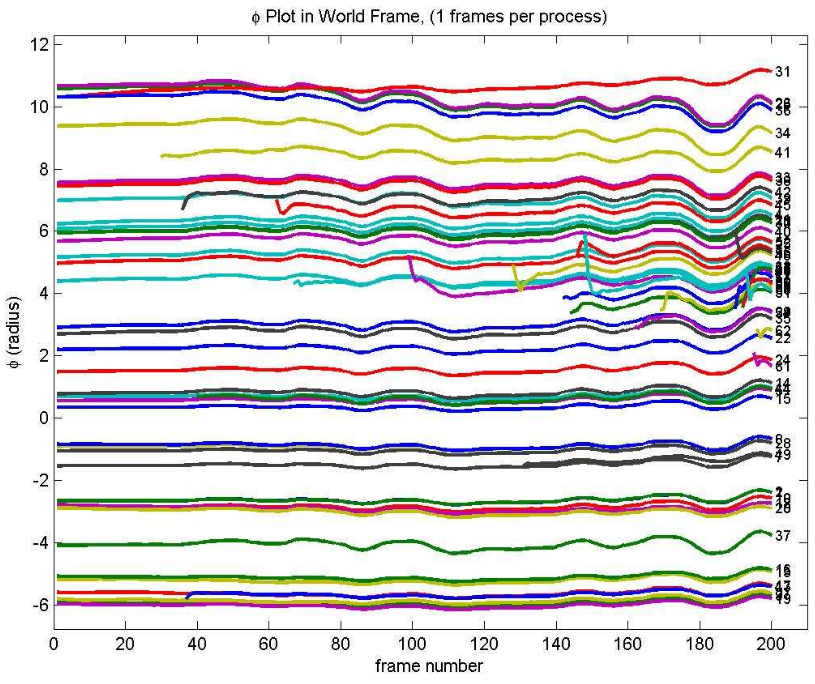


Figure 11 plot of angle phi over 200 frames

maxdrift_phi =1.92

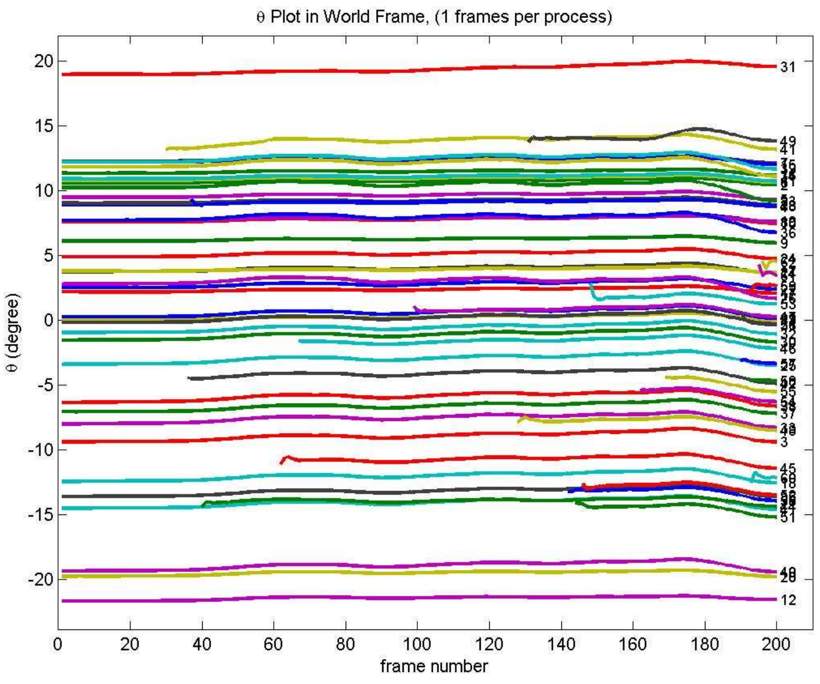
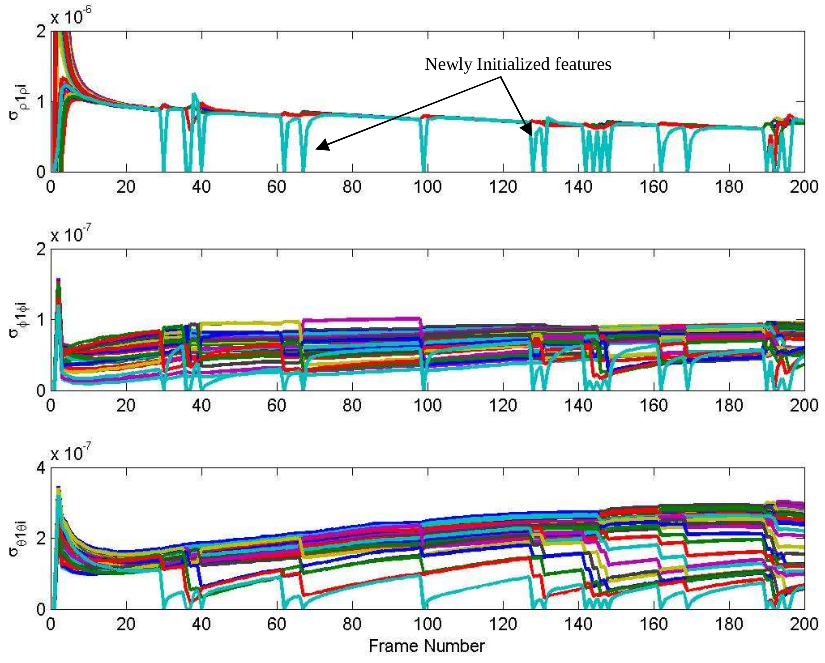


Figure 12 theta plot over 200 frames

maxdrift_theta = 1.5862

Figure 11 and 12 shows the tracking of phi and theta over the 200 frames processed. These two values are calculated from feature coordinates on the image plane on the first frame, and should experience very little change thereafter. Similar to xi, yi, and zi, value of theta and phi is influenced by the aircraft totational motion. The plot of phi is highly correlated to aircraft pitch angle, and theta is correlated to aircrarft yaw. The change in value for phi and theta is also affected by the feature tracking method, and is discussed in detail in section??

Another character of SLAM using Kalman filter tracking is that the correlation between features grows overtime. This character helps to maintain the stability of features parameters during tracking, ie. One bad measurement doesn't have big impact on the features parameters. To observe whether the examined algorithm carries such character, the correlation of the feature parameter: rho, phi, and theta between the first feature and all other features are plotted in figure ? All parameters are initialized to zero. Over time, all three parameters grows to a positive value which indicates a correlation exists between features. The correlation for rho decreases over time. For phi, the correlation stays unchanged, and for theta, it continues to rise after 200 frames.

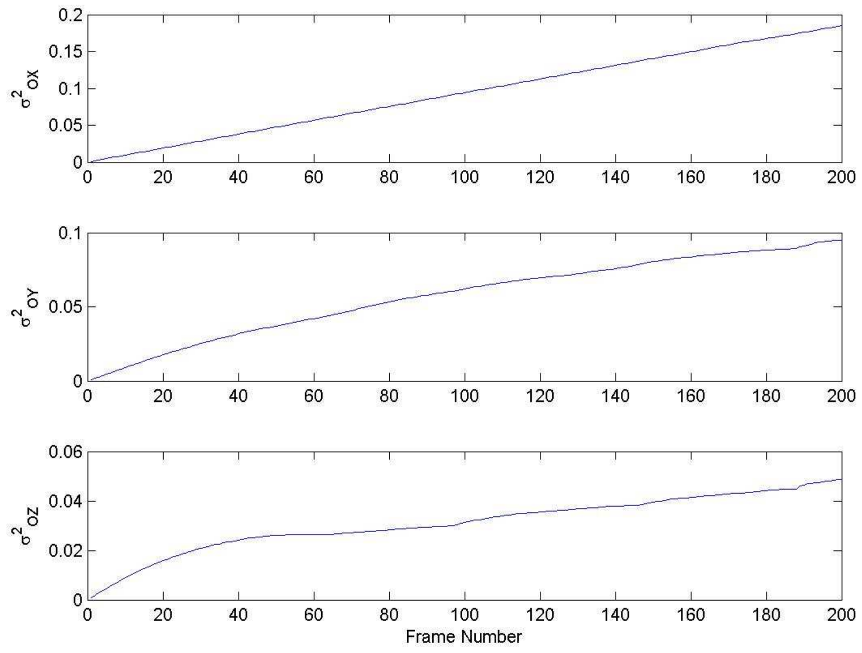


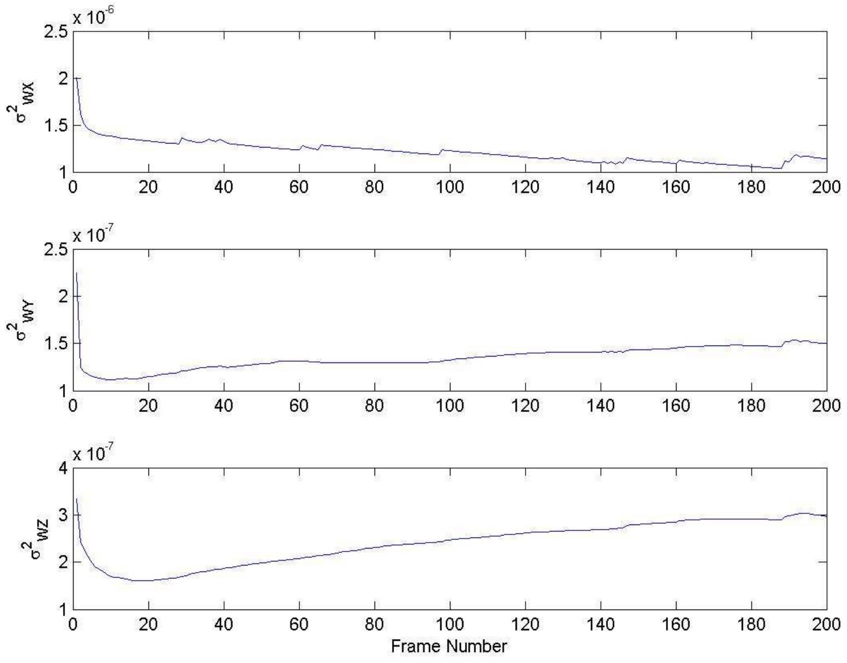
- Covariance between features parameters are initialized to zero
- Inverse depth covariance converged to a positive number, and then decrease slowly as frame number grew.
- Phi covariance grew to a positive number and remain stable
- Theta covariance grew to a positive number and continued to increase slowly as frame number grew.

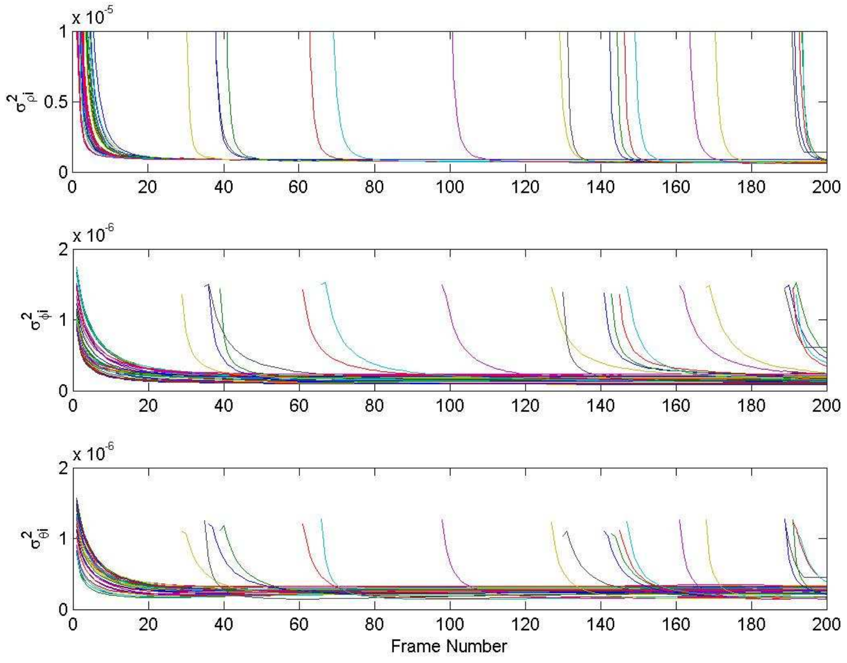
5.2 Consistency

Discuss consistency behavior of the state parameters through examining of the state covariance matrix. Orientation variance should not decrease to zero. If it does, when?

In related work, it has been reported that inconsistency occures when the UAV travel far away from the origin, and caused the filter to diverge. ??? perfomeda analytical analsis on the problem and concluded that inconsistency is caused by evalulating the Kalman equations at the incorrectly estimated position, instead of the real position. [] This usually caused the variance of the state vector to decrease, which in turn, caused an over confident estimation by the filter. Eventually the parameter variance grew to nearly zero, and filter no longer correct the state vector using new measurement, and divergence occurs.




x 10^-6
2.5
2
1.5
1
σ^2_{WX}
0 20 40 60 80 100 120 140 160 180 200
x 10^-7
2.5
2
1.5
1
σ^2_{WY}
0 20 40 60 80 100 120 140 160 180 200
x 10^-7
4
3
2
1
σ^2_{WZ}
0 20 40 60 80 100 120 140 160 180 200
Frame Number


x 10^-5
$\sigma^2_{\rho i}$
x 10^-6
$\sigma^2_{\phi i}$
x 10^-6
$\sigma^2_{\theta i}$
Frame Number

5.3 Reliability

Compare the estimates to ground truth and discuss the accuracy of the results

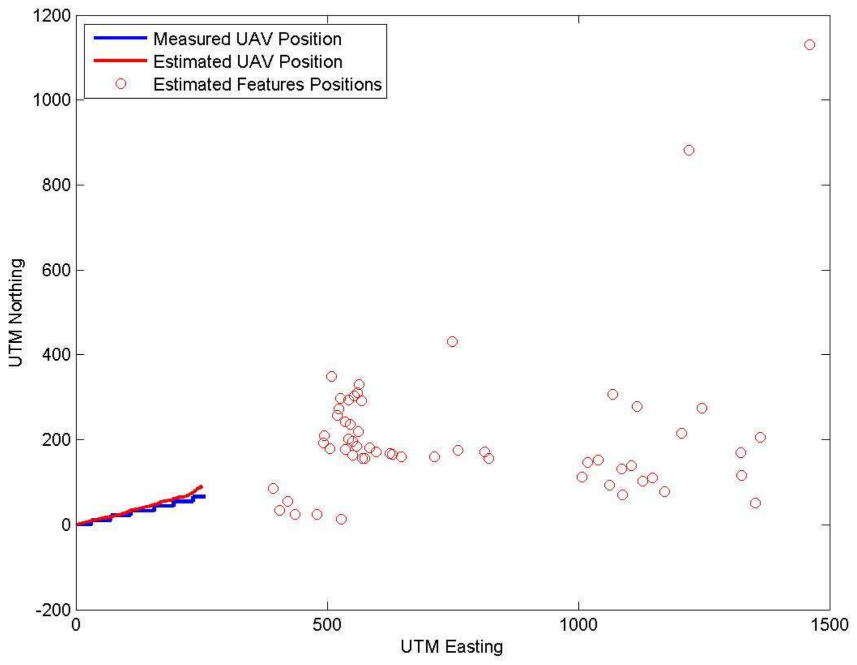


Figure 13 UAV Flight Path vs. GPS

driftUAV = 24.7471 meters

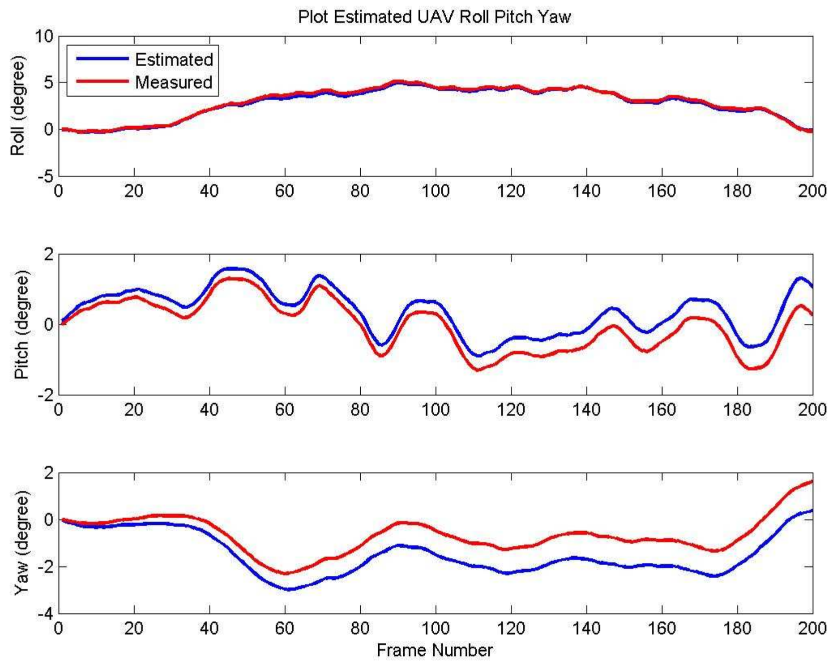
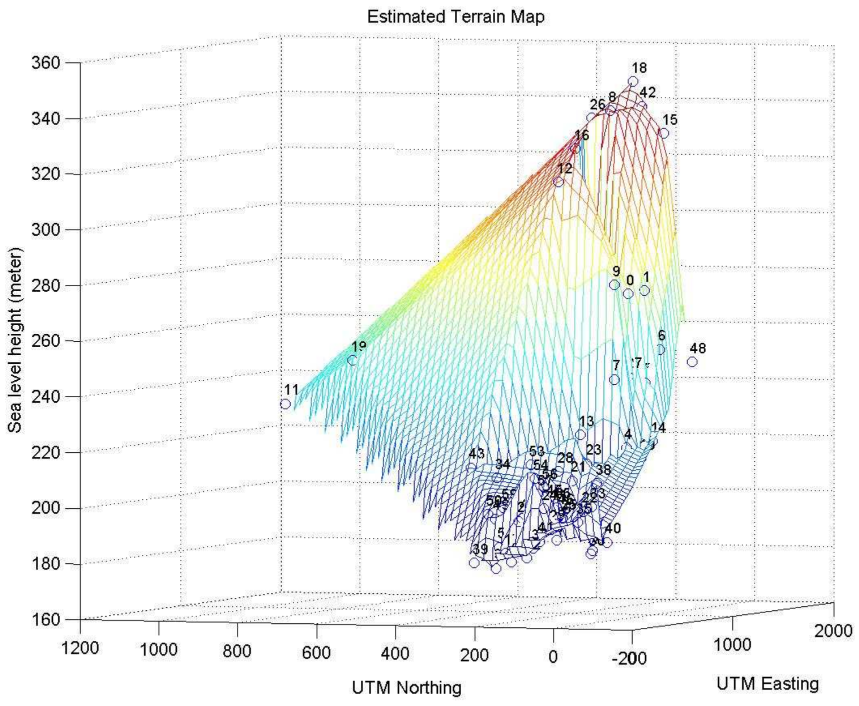


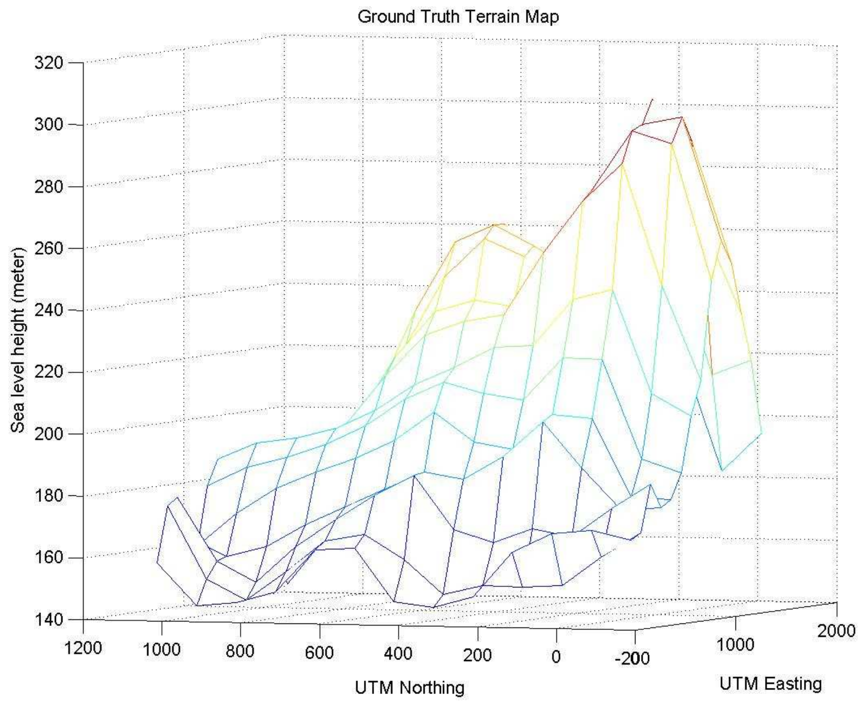
Figure 14 UAV orientation

pitchdrift = -0.7952 degree

yawdrift = 1.2247 degree



Estimated Terrain Map
Sea level height (meter)
UTM Northing
UTM Easting



Ground Truth Terrain Map
Sea level height (meter)
320
300
280
260
240
220
200
180
160
140
1200
1000
800
600
400
200
0
-200
UTM Northing
1000
2000
UTM Easting

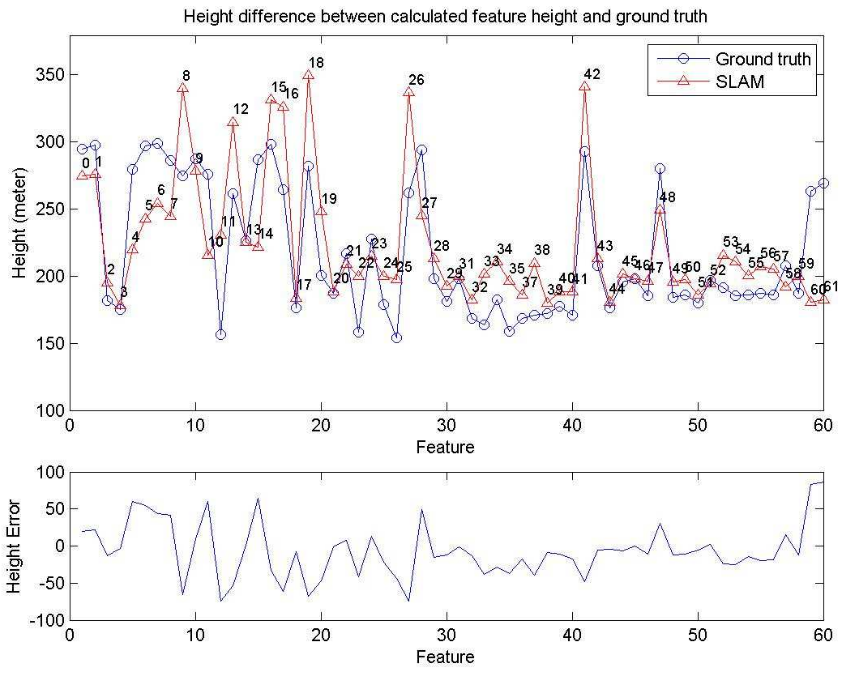
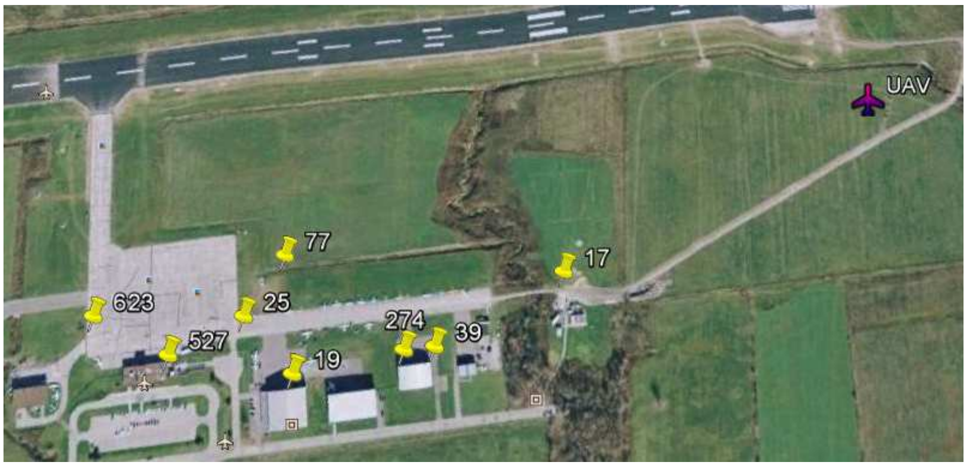


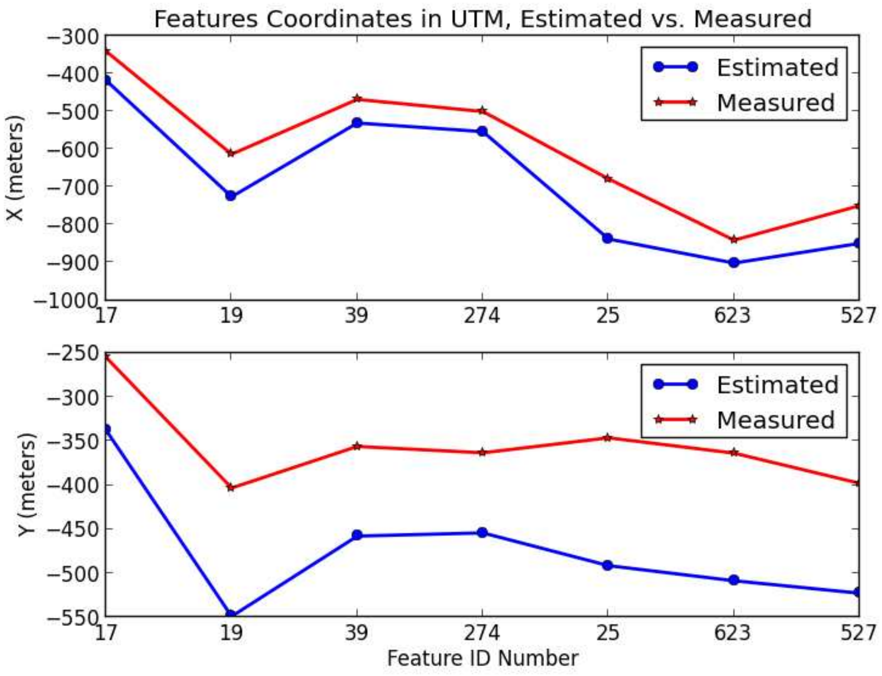
Figure 15 Z coordinate comparison between estimated features and ground truth

Because the survey environment is above forestry landscape, there is a lack of distinct signature to make visual correspondence between the features points and the ground truth map. To make numerical comparison between the estimated features positions and the ground truth, some assumptions must be made. Based on the observation that feature distance is fairly stable once converged, and angle θ is more stable than angle ϕ , the feature coordinate in X and Y is used to find the corresponding position in ground truth map, and coordinate in Z is compared for accuracy.

Figure 15 shows the Z coordinate comparison between the estimated feature and the corresponding data point on ground truth map. The estimated feature height generally follows the pattern of the ground truth. For features located in lower ground, there is a general offset between the estimated height and the ground truth. (Need to process a few different pieces to identify the error pattern). The statistic on the height error is

Airport Landing Video: (Add another picture circling out the feature location in image)





Mean Error on X axis = 89.347 meters

Mean Error on Y axis = 119.428 meters

5.4 Impact of the Outliers

Does the correlation between features help filtering out the outlier measurements?

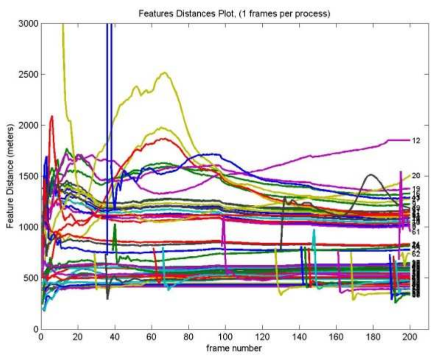
Does the outliers have a negative impact on the overall map quality, and how much?

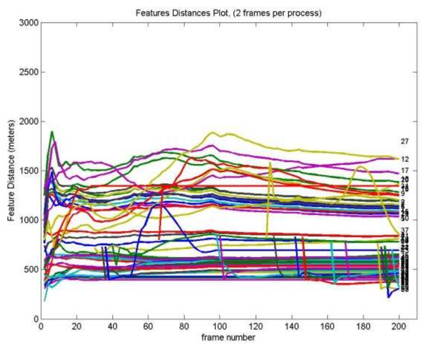
Long term correct measurements help maintaining the system when temporary incorrect measurement occurs. Temporary incorrect measurement can have a negative impact on the overall map quality. Based on properties Consider to delete section

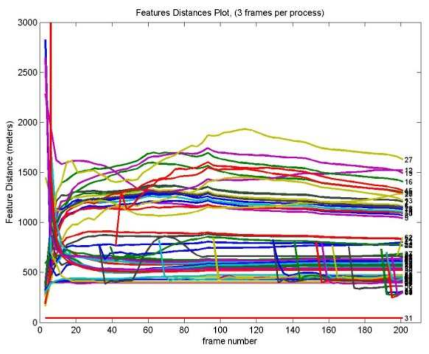
5.5 Use of IMU Improve Vision Tracking Performance in Harsh Flight

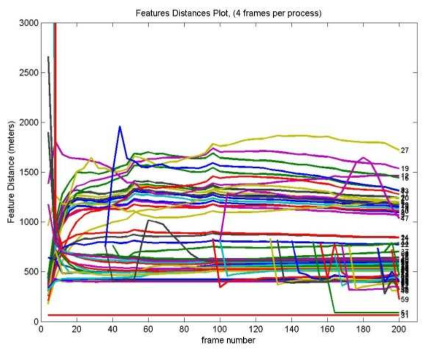
Collect image pairs that shows tracking improvement when using IMU data for prediction. Priority: low.

5.6 Variable number of Frame per Process









Feature Height Accuracy

	1 fpp	2fpp	3fpp	4fpp
mean	-6.6648	-9.6131	-9.4595	-8.6598

std	36.8959	38.9285	41.8112	45.4366
max	86.3177	115.2585	118.5528	132.6752
min	0.1958	0.0979	1.4907	0.7791
# of features	60	58	62	60

No advantages seen on the accuracy. Extract feature at the center (exclude mountain edge) and re-collect data to see if result shows any patter.

Parameters Drift:

	1fpp	2fpp	3fpp	4fpp
xi	2.2102	1.0847	0.7589	0.6668
yi	18.0976	12.1789	12.4523	12.9552
zi	18.4479	13.7087	8.9254	10.5766
phi	1.9168	1.1715	1.2806 (exclude31)	1.6892 (exclude31,51)
theta	1.5862	1.6326	1.6184 (exclude31)	1.7026 (exclude31,51)

Need to check plotting algorithm. Should not have big variation for phi and theta plots.

UAV DRIFT:

	1fpp	2fpp	3fpp	4fpp
GPS	24.7450	19.1648	16.8689	13.9337
Pitch	-0.7952	-1.0177	-1.2818	-1.5072
Yaw	1.2247	1.6169	1.8143	1.9302

Multiple frames per process improves the UAV translational parameter estimation, but offers no advantage in improving the rotational parameters drift.

6 Chapter: Error Analysis via Simulation

There are many factors impacts the accuracy in UAS localization and feature position estimation, and they can be sorted into three main categories:

1. Noise in system intrinsic parameters. The system intrinsic parameters includes
    - Camera intrinsic parameters
        - Coordinate of optical center on image plane $[c_x, c_y]$
        - Scaling factor to project feature in 3D world to image plane $[f_x, f_y]$
        - Lens distortion parameters $[k_1, k_2, p_1, p_2]$
    - Image resolution.
    - Accelerometer bias
2. Error introduced by LK tracking algorithm. Reliable vision tracking is an entire field of research in itself. Pyramid implementation of Lucas-Kanade (LK) tracking is used in this work, and there are a number of factors contribute to its performance. Firstly, LK tracking algorithm tracks features by comparing the intensity of a window of the image centered at the feature coordinate in image from one frame to another. The searching process terminates when the sum of error on the windowed image intensity is lower than a value set by user, or when iteration of search has reached a maximum number set by user. Secondly, as scene evolve from frame to frame, the initial feature appears differently from frame to frame as the viewing distance and angle is different. Thirdly, sudden intensity change in the image sequences significant noise in the tracking. In outdoor setting, intensity change can be introduced by many factors, such as changes of sky area in a image, sun glare, UAV enters or exits cloud shades, or camera auto-adjust its shuttle speed, etc.
3. Error caused by the SLAM algorithm itself. The algorithm estimated features coordinate through a model that represents the relation between UAS location, feature location and UAS motion. As the model is non-linear, the linearization process introduces error into the result.

To better understand the impact of the factors listed above. A simulation is performed to examine the impact item 1 and item 3.

The simulator first generates a 3D point cloud ranging from 100 meters to 3000 meters from the camera. At each frame, the coordinates of the 3D points are first transformed to the new camera frame using the measured UAS motion. Next, the 3D points are projected to 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.

6.1 An Ideal Case

First of all, understanding the algorithm's performance under no noise, or nearly no noise condition provide a solid ground for the analysis later on. This simulation shows how much error the model itself generates under the most basic flying condition, which is moving forward at constant speed. The low noise environment is configure as such,

- UAS is moving forward (X axis) with constant speed at 60 knots.
- Y axis and Z axis translational movement are limited to white noise with standard deviation of 0.08 meters and a mean of 0.
- UAS rotation are modelled by white noise with standard deviation of 0.01 degree and a mean of 0
- No error was introduced due to image digitization. (i.e. the projected feature position on image plane was not digitized to any sensor resolution)
- No error was introduced from camera model mismatch. (i.e. camera model used by simulator is exactly the same as the one used by CC_EKF_SLAM algorithm.

6.1.1 UAS Localization

The estimation of UAS coordinate and orientation is first analyzed, as these estimates are directly used to perform transformation between camera and world frame. Figure 16 below plots the UAS translation and rotation against video frame number. The ground truth and estimated value are plotted in blue and green lines respectively. The error defined by $Estimated - Ground\ Truth$ is plotted in red line. Under a simple forward only motion, the algorithm tracked the UAS status quite well, with error on translational motion less than 1 cm and error on rotational motion less than 3e-3 degree.

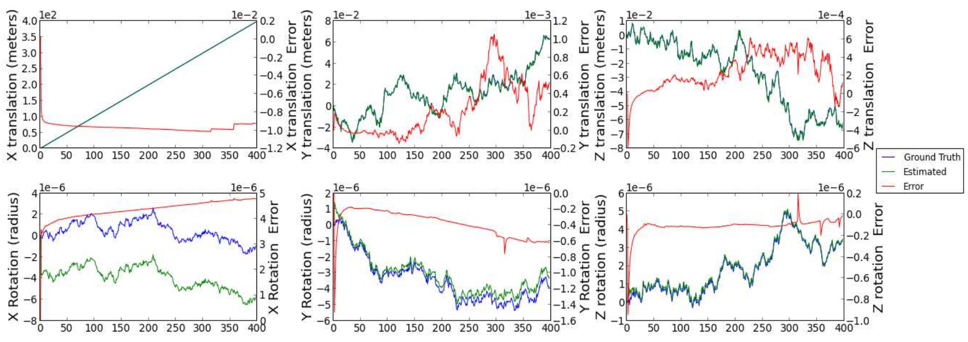


Figure 16

6.1.2 Features Mapping - Convergence and Accuracy

Figure 17 shows feature parameters $[d, \phi, \theta]$ (where $d = \frac{1}{\rho}$) plotted against frame number for the first 50 frames. The feature depth d for all features converged within 3 frames; elevation-azimuth angles $[\phi, \theta]$ stay almost constant after initialization. A more detail graph can be seen from the error convergence plot for these parameters (Figure 18) which shows the tracking error of these parameters for 400 frames. The error of feature distance d continues to approach zero as the tracking continues. $[\phi, \theta]$ show small drift within $+\frac{\square}{-0.0002}$ respectively. However, as tracking continue into later frames, error of $[\phi, \theta]$ gradually grew bigger. The resulting error for feature coordinate in world frame represented in standard Euclidean XYZ parameterization is plotted in ??. The features positions errors in world frame converge to zero as tracking continues. During the process, certain features moved out of the FOV, and therefore its position estimation remained unchanged since then. At the end of the 400 frames, the x axis position error of the feature reduced to +/-0.2 meters; y and z axis error reduce to +/-0.02 meters

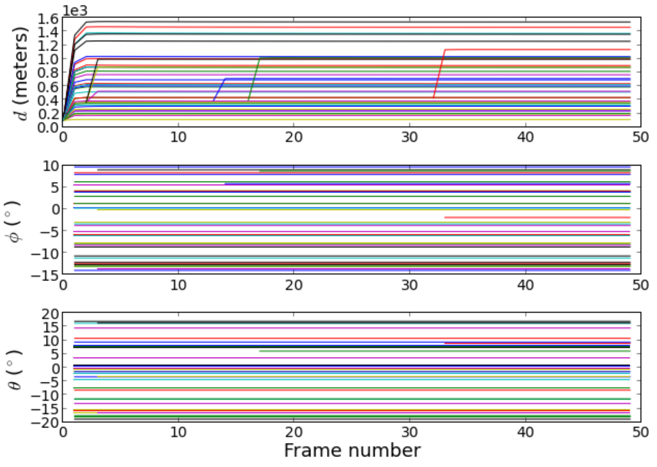


Figure 17 Feature parameters convergence over 50 frames

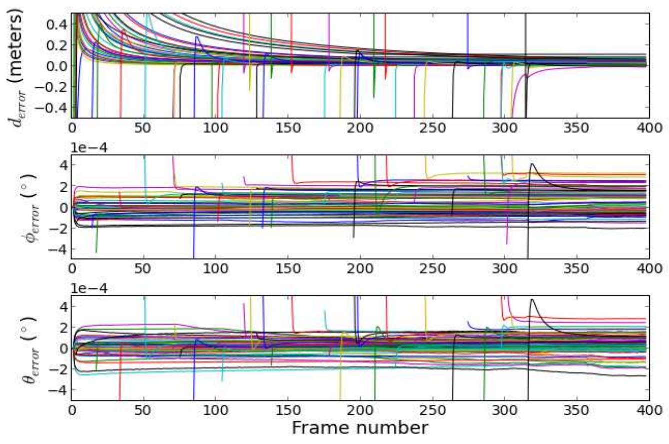
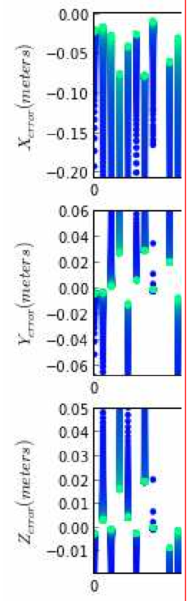


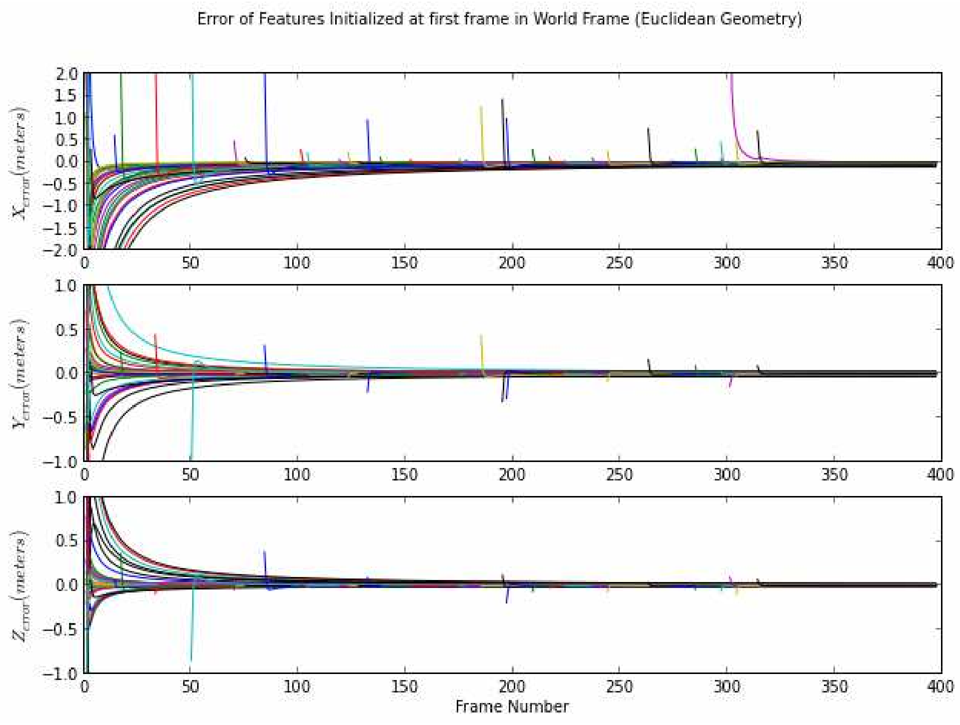
Figure 18 Error of feature parameter convergence over 400 frames

Comment: delete

Deleted: Figures below show features error plotted against feature ID (on the left), and against frame number (on the right). At some feature ID, there is no data point plotted. This is because those features were filtered by the minimum tracked frames criteria. The face color of the markers represents frame number. It start from dark blue at frame 0, and ends with light green for frame 398. We can see in both plots, the accuracy of feature position are quite good, with estimation error on all three axis less than 0.1 meter.



Deleted:



6.2 Effect of UAS Motion

The simulation result from UAS forward travel shows that the CC_EKF_SLAM algorithm does feature tracking and self localization quite well under simple UAS motion. Next, the algorithm is tested with a more complex and realistic scenario. A series of motion is added to the simulation in addition to the forward motion. The remaining 5 types of maneuver are added one at a time. These maneuvers are: translation on Y, translation on Z, rotation on X, rotation on Y and rotation on Z. Each motion is modelled by a sine wave with frequency at 1Hz, and variable amplitude. For translation on Y and Z axis, the sine amplitude varies from 1 meter to 19 meters with 2 meters increments. For X, Y, and Z axis rotation, the amplitude varies from 0.001 radius to 0.018 radius with 0.001 radius increment.

6.2.1 UAS Localization under Motion

Figure 19 shows the UAS localization error statistic under translation motion on Y and Z axis. Figure 20 shows the UAS localization error statistic under rotation motion on X, Y, and Z axis. The blue dots mark the mean value μ of the error throughout the tracking, and the error bars mark the standard deviation σ.

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

On the other hand, rotation motions have a big impact on the accuracy of localization. Rotation on X axis has small effect on the accuracy of UAS position and orientation estimate. No obvious increase on mean and standard deviation of the error can be observed. Rotations on Y and Z axis yield significant error increase on the position of the UAS.

- Rotation on Y axis increases the UAS X position mean error, as well as the standard deviation. For Z positioning of the UAS, the mean error stays zero, but standard deviation increase dramatically.
- Same thing happens to the X and Y positioning for rotation on Z axis.

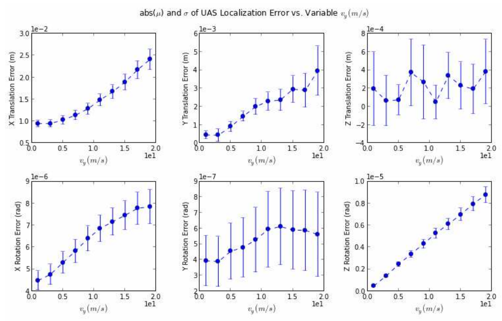
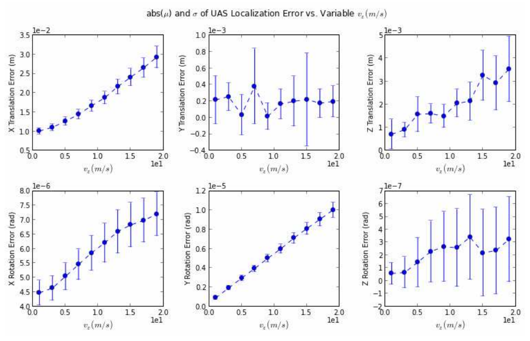
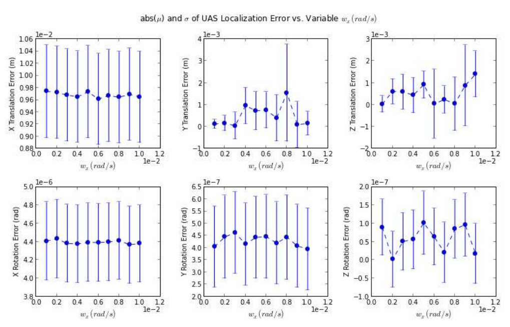
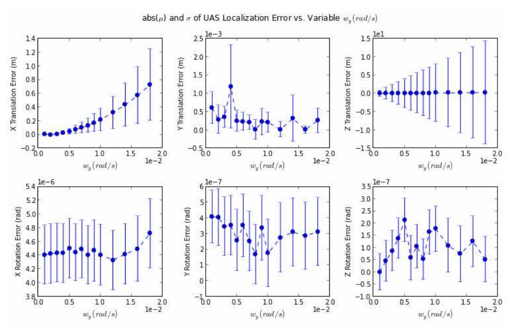


Figure 19







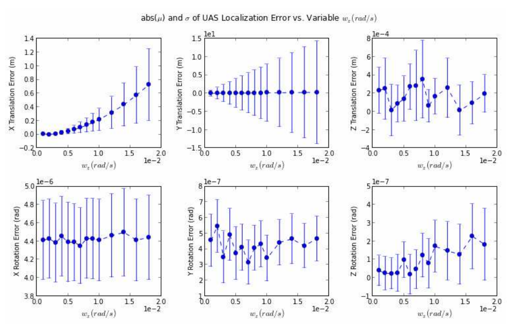


Figure 20

To understand how rotation motion affect the UAS localization estimation, the UAS position on X, Y, Z in world frame are plotted below (Figure 21) with rotation amplitude on X, Y and Z set at 0.01 radius. With rotation on X axis, the position error of UAS shows some oscillation. The oscillation magnitude remains small (in the scale of millimeters) and around zero. For rotation on Y and Z, the situation is very different. Both rotation motions caused the UAS position error to oscillate with the oscillation amplitude increasing (diverging) as tracking goes on. The X position error of the UAS increases in positive value with both rotation on Y and Z, but the most significant impact happens on the Z position (for rotation on Y) and Y position (for rotation on Z), with error reaching 20 meters at the end of the video sequence. With rotation rate increases (amplitude of the sine wave), the rate of error diverging from zero also increases, hence, resulting in an increasing error standard deviation in error statistic plots. This simulation result suggests that **CC_EKF_SLAM algorithm is very sensitive to rotation on Y and Z axis**.

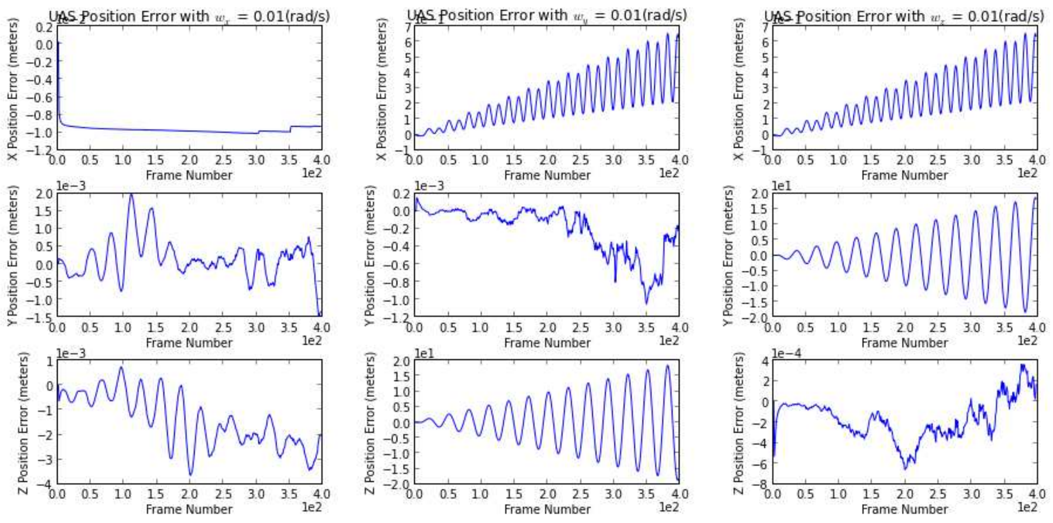
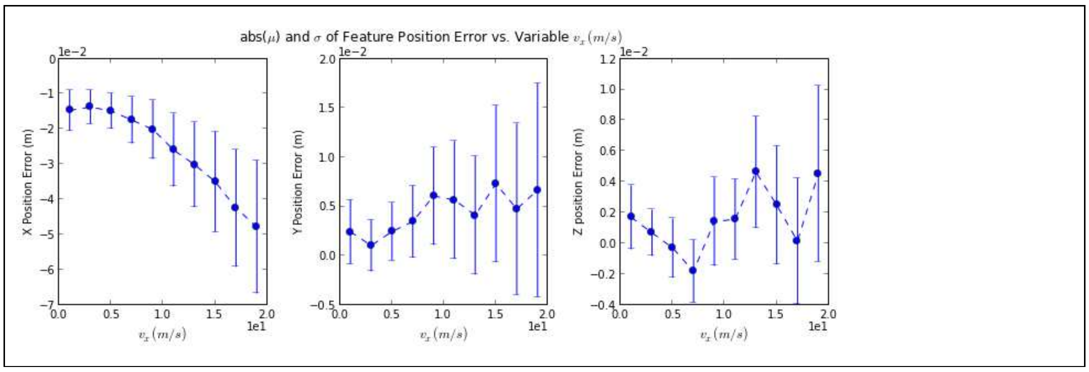


Figure 21

╬

6.2.2 Feature Mapping Accuracy under Motion

Feature mapping error statistic are drawn from the feature error at last frame, since features error converge to zero as tracking goes on. Figure 22 shows the feature mapping error statistic with translation motion on Y and Z axis. Translation motions increase both the error mean and standard deviation, but not by much. With the motion maximum amplitude ranging from 1 meters to 19 meters, the increases of features position error mean and standard deviation are both in the scale of centimeters.



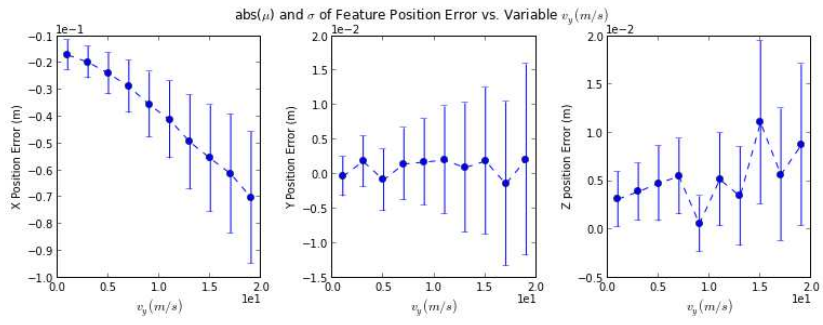
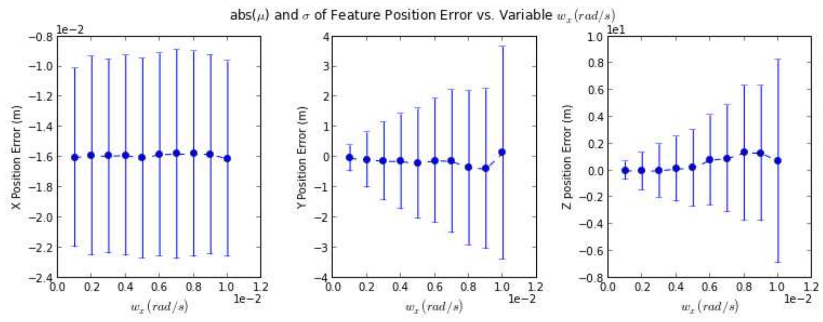


Figure 22

Figure 23 shows the feature mapping error statistic for rotation motion on X, Y and Z axis. With maximum rate of rotation ranging from 0.001 rad/frame to 0.018 rad/frame

- Rotation on all three axis yields significant error increase for feature position estimation
- X axis rotation causes increase on standard deviation of Y and Z axis feature position error, and by similar amount. The error are in scale of meters with maximum rotation setting.
- Y axis rotation causes increase on mean and standard deviation of X and Z axis feature position error. Z axis feature position receives the biggest impact with error in the scale of hundreds of meters with maximum rotation setting
- Z axis rotation impacts on X and Y axis feature position in a similar way as the Y axis rotation.



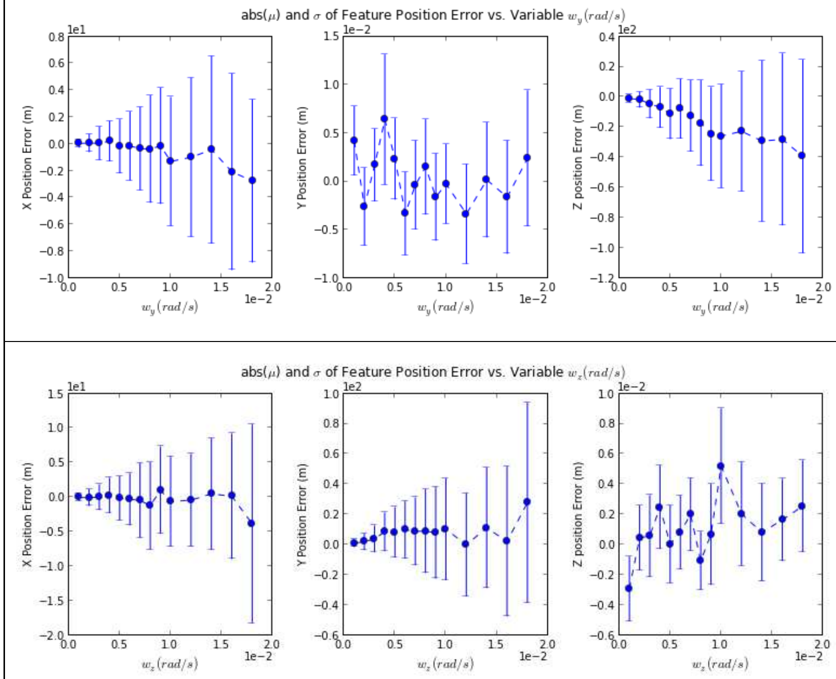


Figure 23

Figure 24 to Figure 26 shows the features position error at the last tracking frame for all three type of rotation motion. The errors are plotted against feature ID and it revealed some more characteristic of the features mapping errors.

- With rotation on Y and Z, the tracked features easily went out of FOV. This can be observed from total number of features went from 80 to over 110 with Y axis rotation setting varied from 0.001rad/s to 0.018rad/s. This caused frequent addition of new features.
- Features added after first frame has much bigger error than features added at first frame. At 1st frame, 40 features were added to the filter. Error plots from Y and Z axis rotation both shows that major feature mapping error came from features added after the 1st frame

with ID bigger than 40.

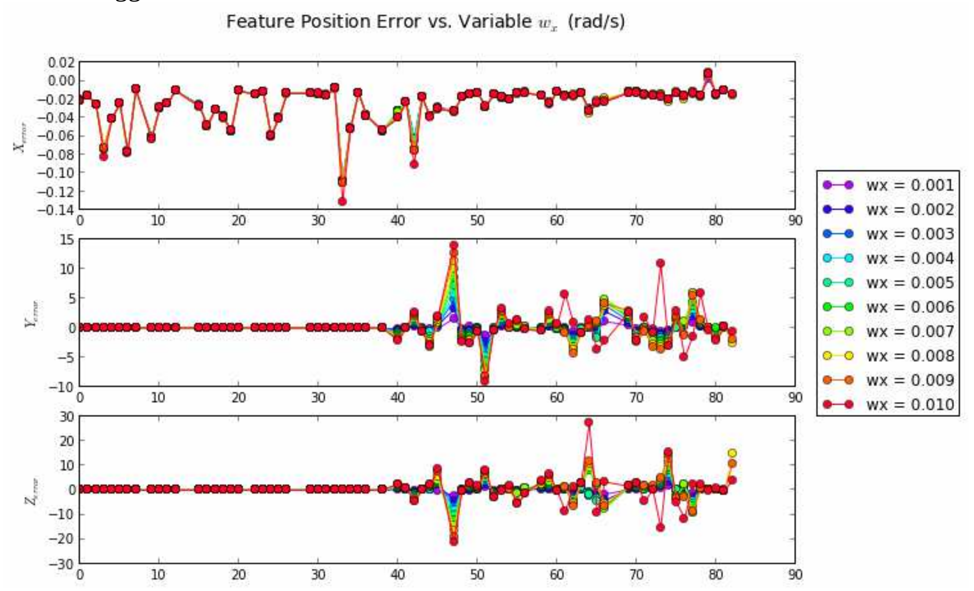


Figure 24

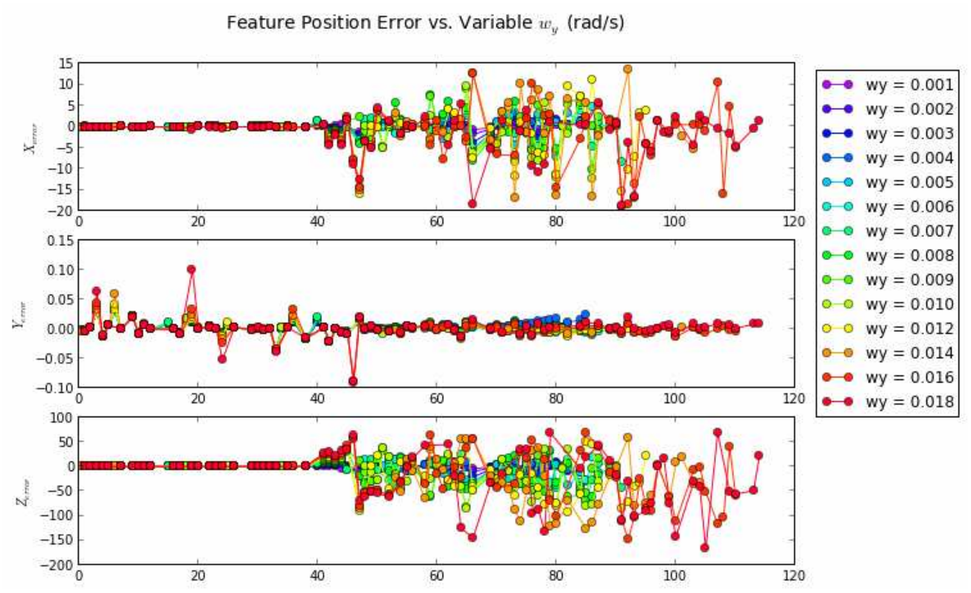


Figure 25

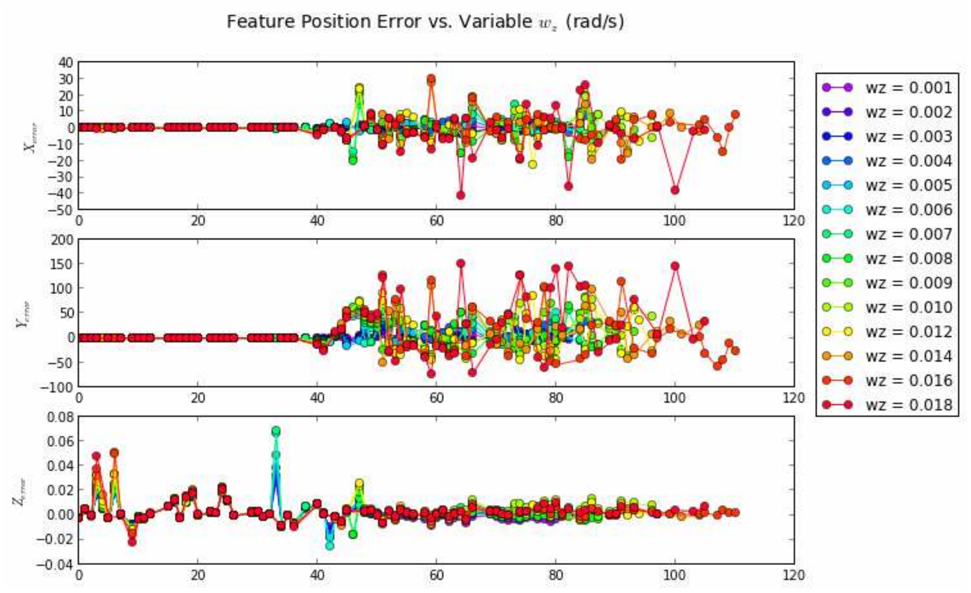


Figure 26ie

To investigate how does rotation motion results in bigger error on features added after first frame, feature parameters error (converted to world frame) with $w_y = 0.01$ are plotted in Figure 27. It is found that the most significant error happen to parameter φ which is the feature elevation angle. This angle has the same definition as rotation angle around Y axis. The second contributor is z_i (the Z axis coordinate of the feature initialization point). The both parameters have an offset error at initialization, and were never corrected throughout the tracking.

Figure 28 shows the φ error at initialization in camera frame and world frame. The blue line shows the error in camera frame. The red line shows the error in world frame transformed using the estimated UAS position and orientation. It is clear that it is the transformation process that introduced the offset error in φ. Offset error in z_i is due to the same reason. Recall that with Y axis rotation, UAS localization estimate has biggest error in X and Z axis coordinates estimate. Features initialized at first frame don’t carry any offset error is because the transformation process is using the same parameters in both way. During tracking, these features are transformed to the new camera frame using the estimated UAS position and orientation. These are the same estimation being used to transform feature position from camera frame into world frame. Therefore, although the UAS localization estimations are different from the ground truth, feature initialized at first frame were not affected. To conclude, the major contributor for feature mapping error came from error in UAS localization estimation.

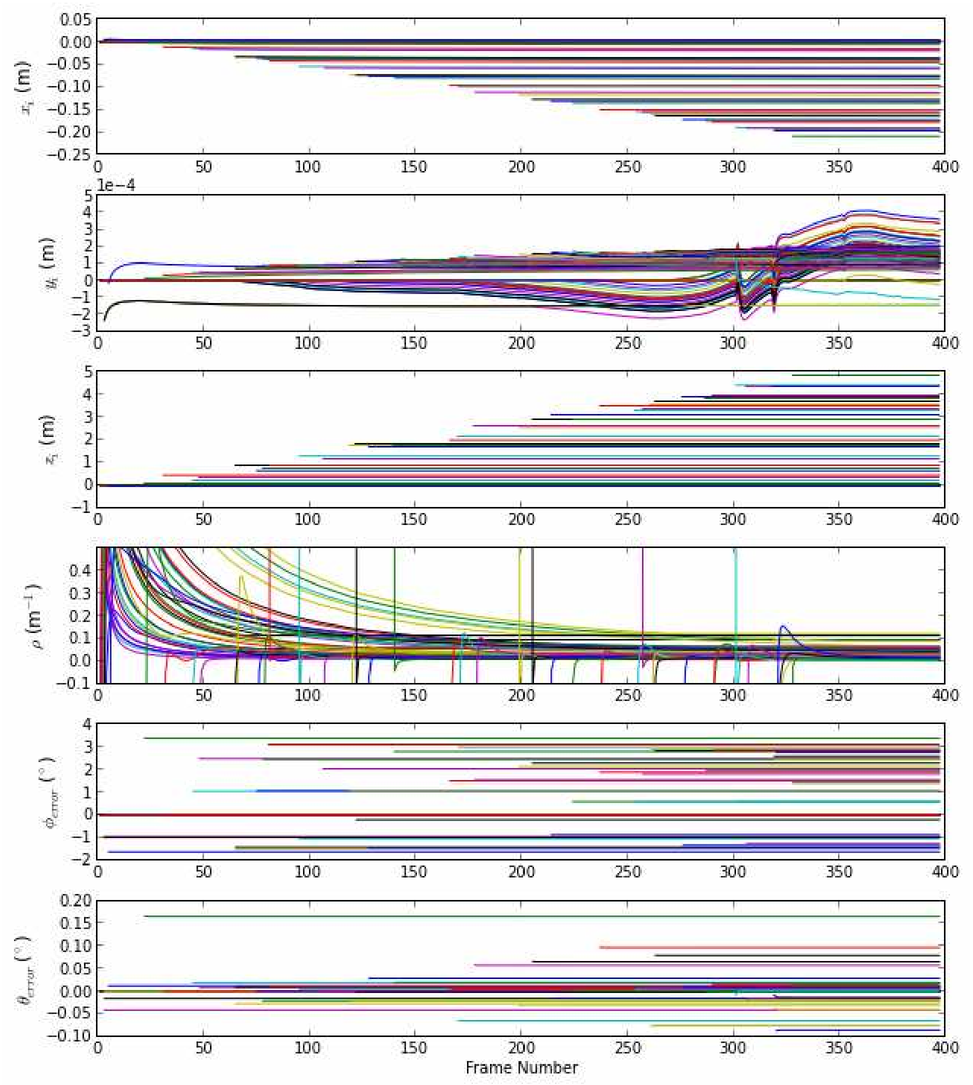
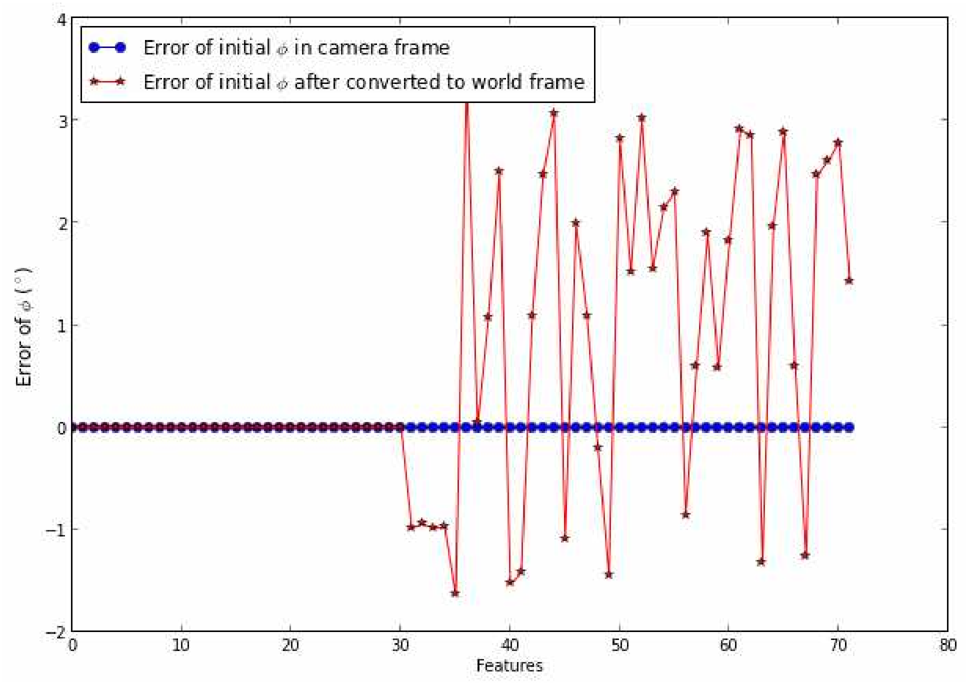


Figure 27



Error of initial ϕ in camera frame
Error of initial ϕ after converted to world frame
Error of ϕ ($^\circ$)
Features
4
3
2
1
0
−1
−2
0
10
20
30
40
50
60
70
80

Figure 28

6.3 Camera Intrinsic Parameters

This section summarizes the impact of inaccurate camera parameters estimations. Error on camera intrinsic parameters is simulated by using different values for camera models. One model is used in the simulator to project 3D points onto image plane, and the other is used in the measurement model of CC-LK-SLAM. c_x , c_y , f_x , and f_y are simulated individually and distortion parameters $[k1, k2, p1, p2]$ are simulated as a group. Using the calibrated camera model (see section ???) as a base model, c_x , c_y , f_x , and f_y in simulator camera model varied from -50% to 50% of the base model. Distortion parameters varied from 0% to 140% of the base model.

6.3.1 Effect from (c_x, c_y)

Figure 29 and Figure 30 show an over view of UAS localization error statistics with incorrect estimates of (c_x, c_y).

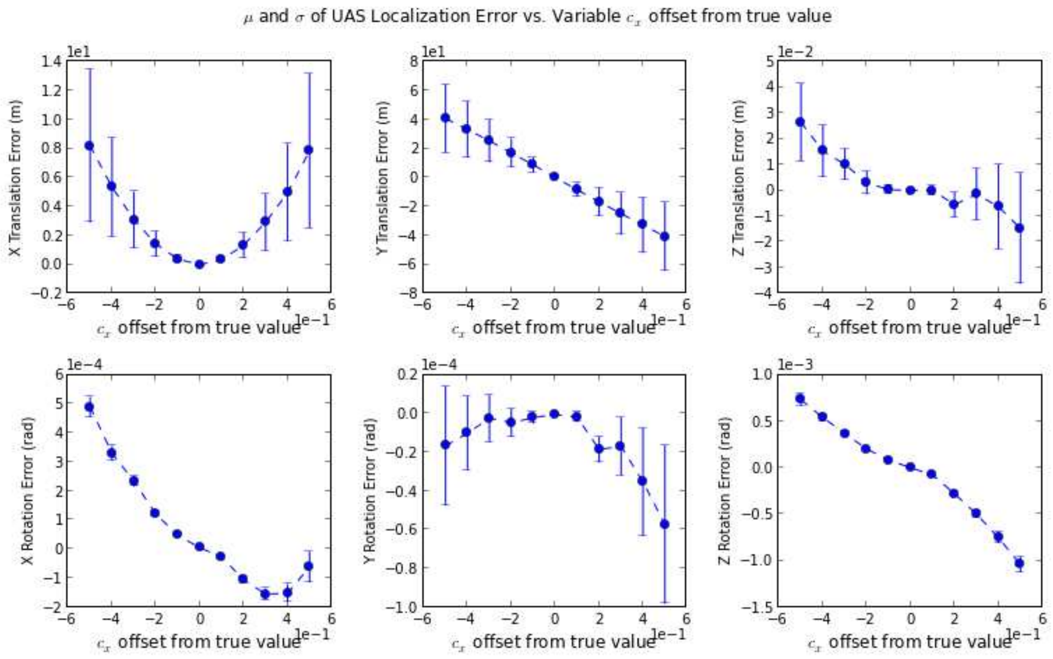


Figure 29

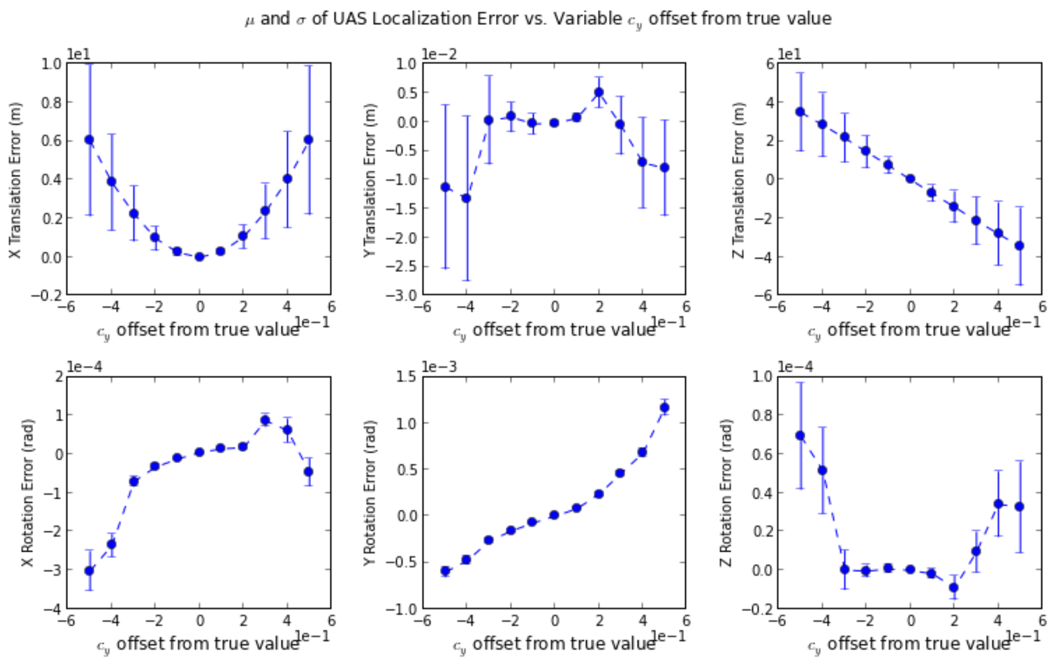


Figure 30

UAS position error is dependent on (c_x, c_y) and time (Figure 31-Figure 32). The UAS position error is diverging (increases in time), and can be modeled by 1st order polynomial function, with the rate of diverging decided by the error of (c_x, c_y). c_x affects UAS position on X and Y axis, and c_y affects UAS position on X and Z axis.

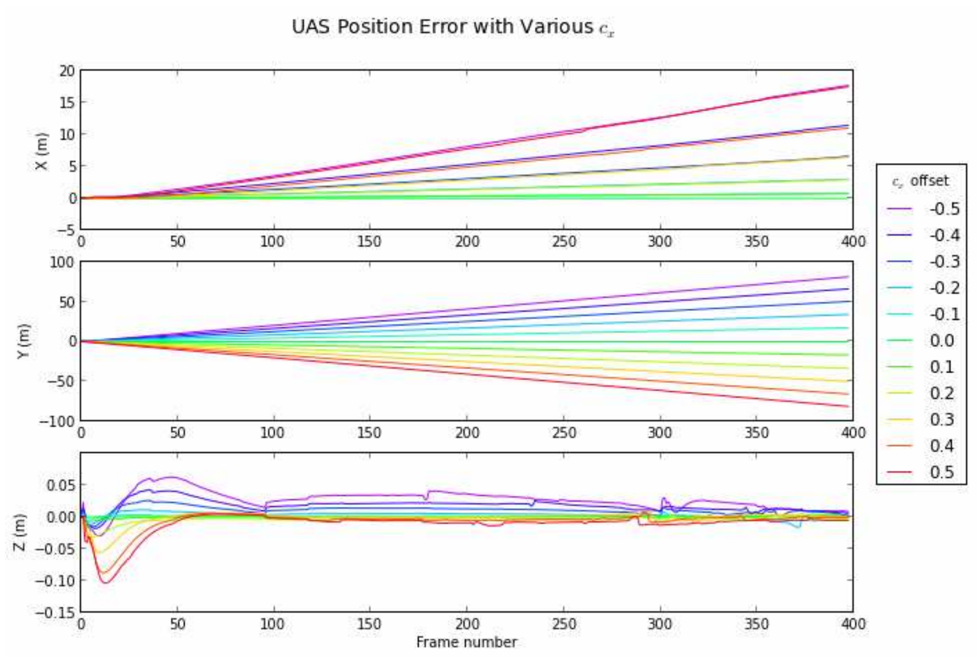


Figure 31

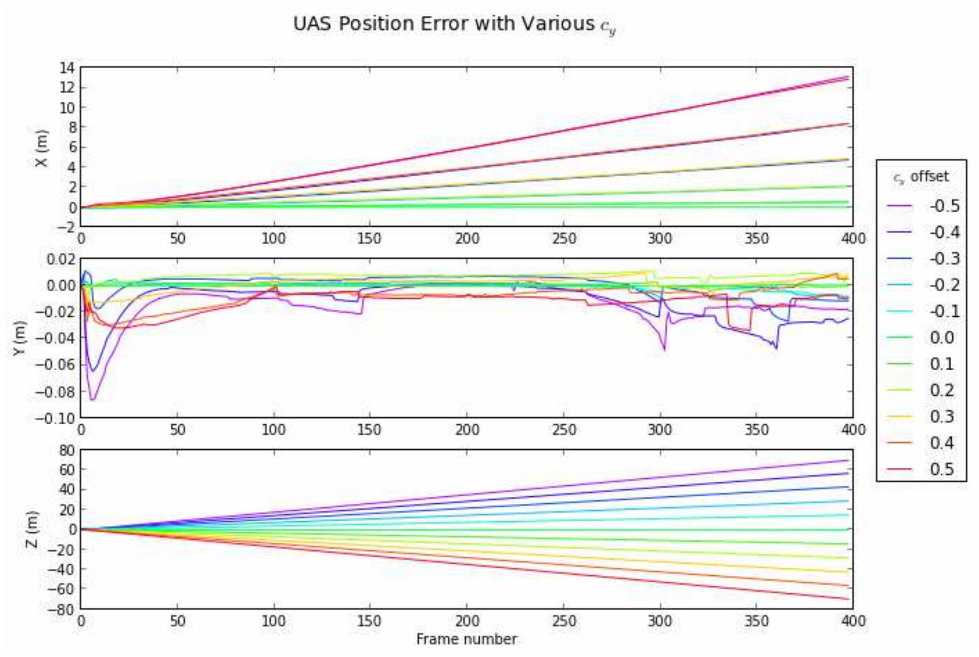


Figure 32

The feature mapping error statistics from incorrect estimate of (c_x, c_y) are plotted in Figure 33 and Figure 34. The following characters can be observed from the plots:

- Incorrect c_x affect feature position on all axis, among which, X and Y axis see the most significant error.
  - o The further c_x deviate from the true value, the further the features appear (positive X axis error).
  - o On Y axis, smaller c_x make feature appear further to the optical axis than ground truth (positive error), and bigger c_x make features appear closer (negative error).
  - o Incorrect c_x also affect Z axis feature position, but by a much smaller amount.
- Incorrect c_y affect feature position on all axis similarly to c_x. Estimates on X and Z axis show most amount of error.

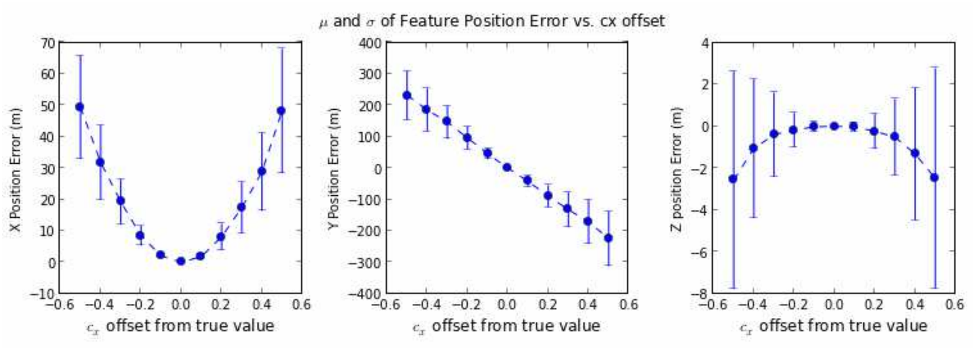


Figure 33 CX offset vs. estimated feature coordinate error

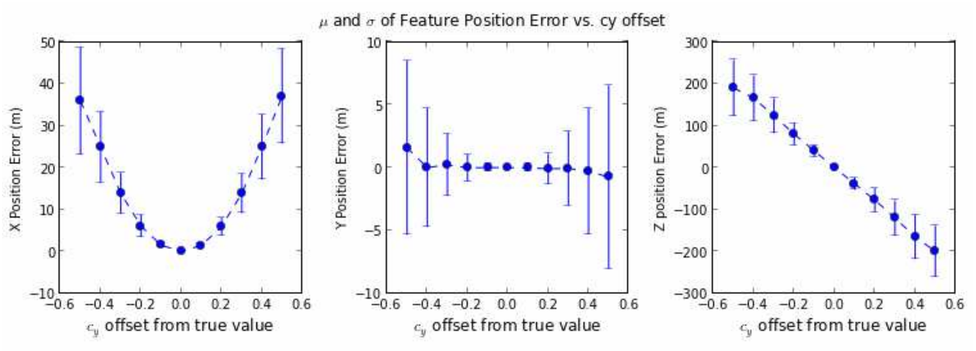


Figure 34 CY offset vs. estimated feature coordinate error

Plotting feature position error as a function of features ground truth positions reveals more information on how incorrect c_x and c_y affect feature mapping. Feature position error is a function of its ground truth position, and c_x (or c_y).

- Error on X axis is proportional to the ground truth position on X. The further the feature is, greater the error. The degree of incorrectness in c_x decide the slope of the error plot, greater the error in c_x, steeper the slope (Figure 35, plot[1,1]).

- Feature position error on Y axis is also proportional to the ground truth position on X with the slope polarity dependent on the polarity of the error of c_x, and error plot slope dependent on the error of c_x.
- Feature position error on Z axis is proportional to the ground truth position on Z, with slope polarity dependent on the polarity of error of c_x, and slope dependent on the error of c_x.

c_y affects feature position similarly to c_x (Figure 36), except feature position error on Z is dependent on ground truth position on X, and error on Y is dependent on the ground truth position on Y.

$(c_x, c_y)(c_x, c_y)$

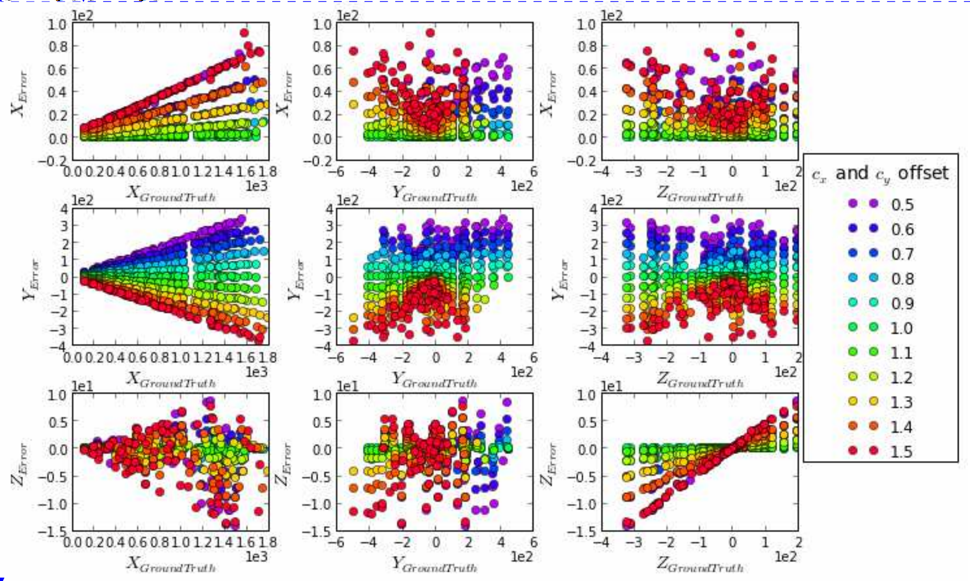


Figure 35

Deleted: With the relationship between feature mapping error and error of

Deleted: identified, it is possible to calibrate previously calculated feature position if a more accurate estimate of

Deleted: can be obtained. ¶

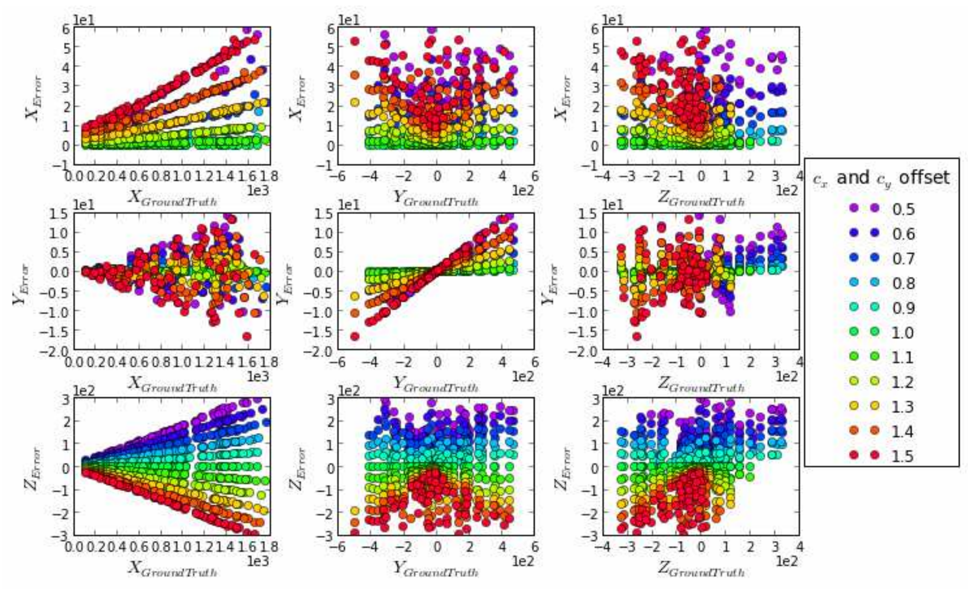

X_{Error}
Y_{Error}
Z_{Error}
$X_{GroundTruth}$
$Y_{GroundTruth}$
$Z_{GroundTruth}$
1e1
1e2
1e3
c_x and c_y offset
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

Figure 36

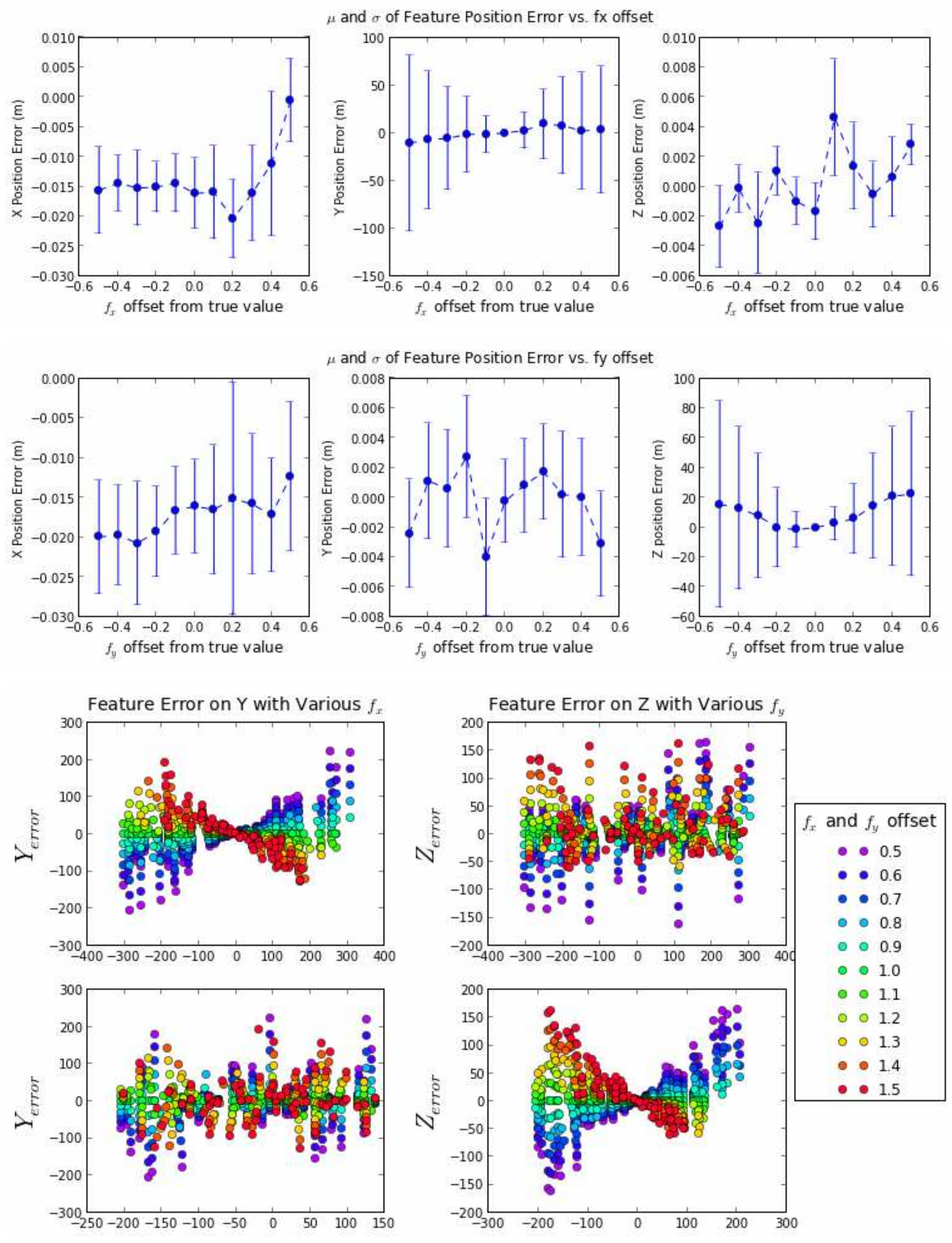
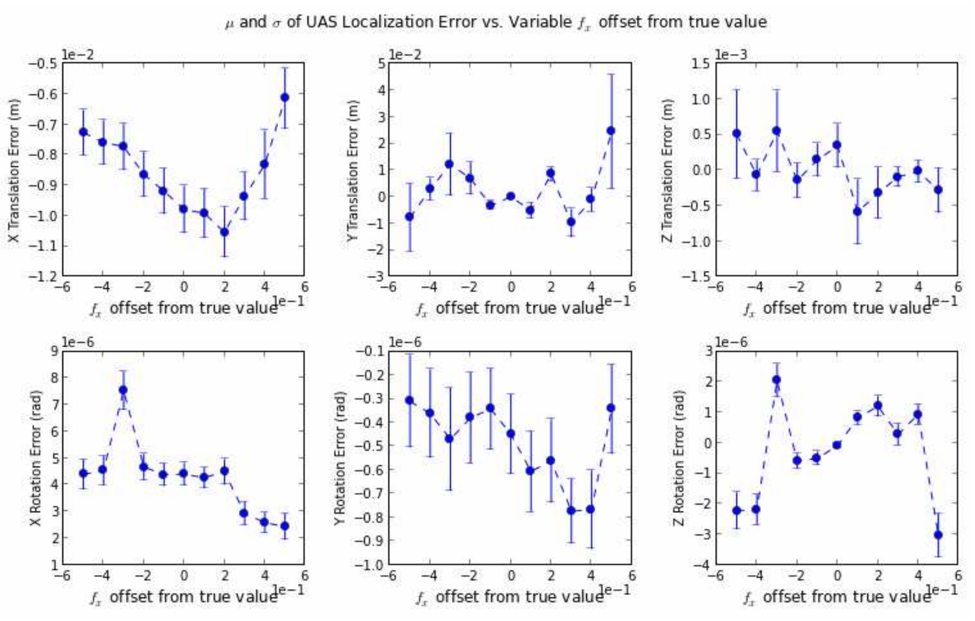


Figure 37 FX offset vs. estimated feature coordinate error



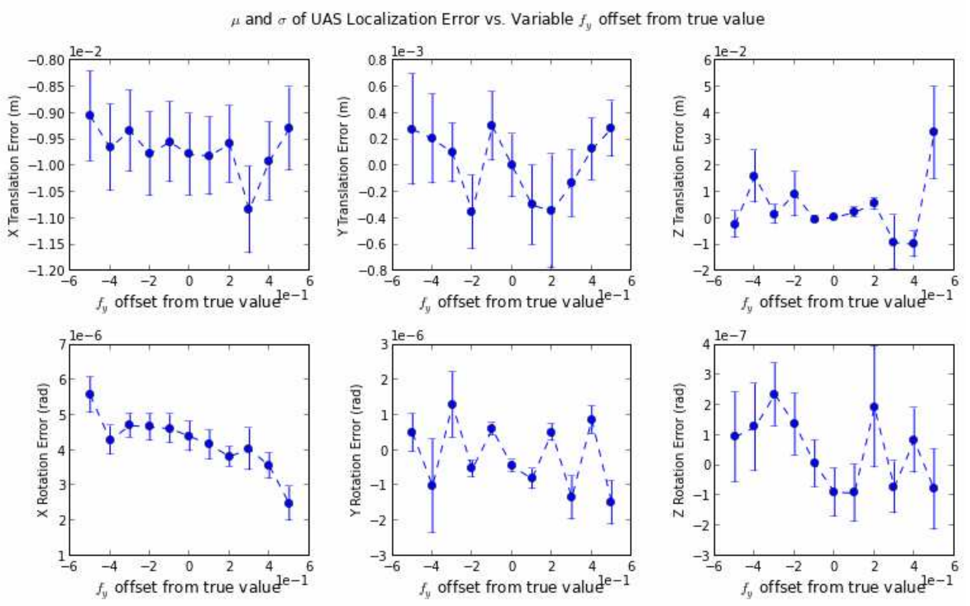
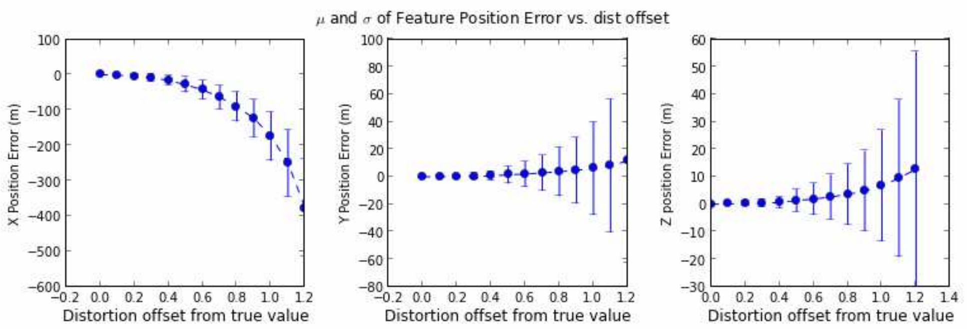


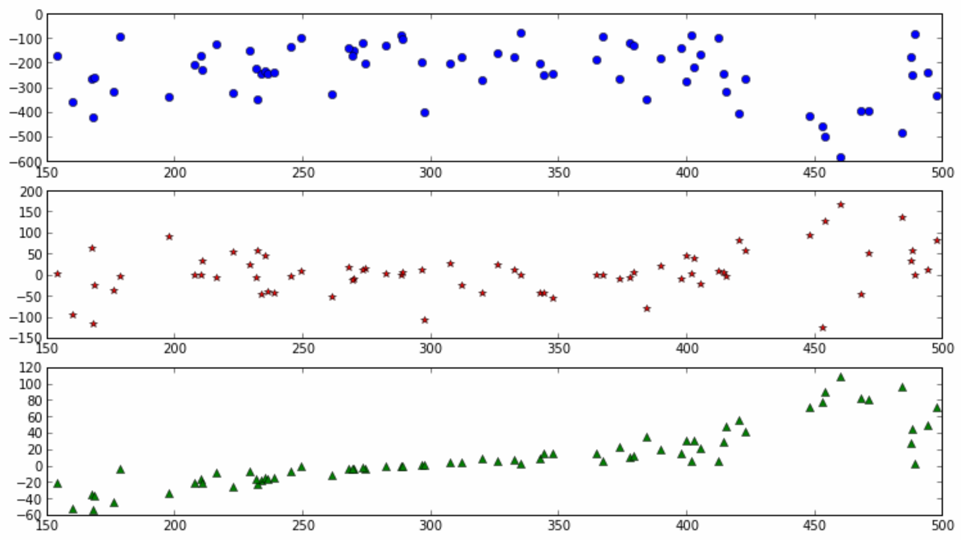
Figure 38 FY offset vs. estimated feature coordinate error

Figure 37 and Figure 38 show the features coordinates error for all the f_x, f_y setting. The following characters can be observed from the plot:

- Deviation of fx has more affect feature position estimation on the Y and X axis; deviation of fy from its true value affect feature position estimation on Z and X axis.

- Deviation of fx or fy on either direction causes X component of the estimated feature position to be smaller than the truth value.
- Negative deviation of fx causes little mean error on Y, (fy on Z), but the mean squared error is equivalent to the amount caused by the positive deviation. This indicates that an offset on fx to negative side is not any better than to the positive side.





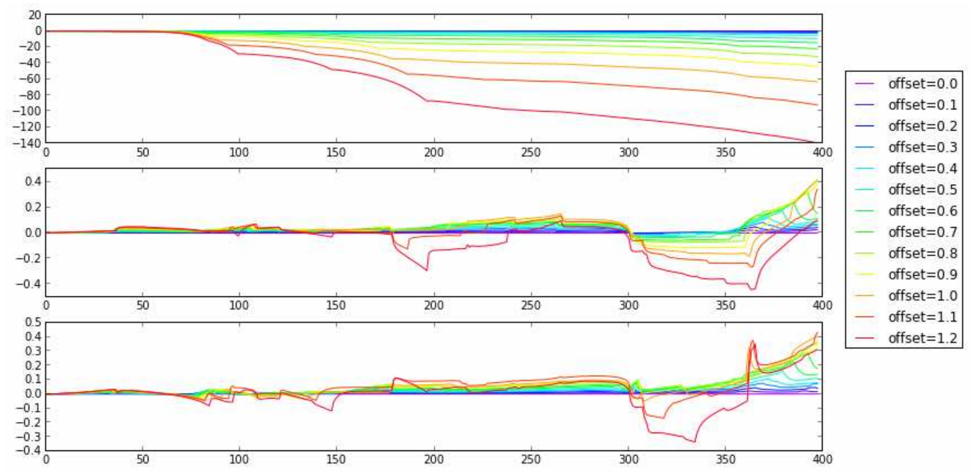


Figure 39

Figure 39 shows the estimated feature position error versus the camera distortion. Since in CC_SLAM no camera distortion is included, the true value of the distortion is [0, 0, 0, 0]. The calibrated distortion parameters are marked at 100% deviation from true value. Therefore, the simulation results at 100% offset shows the average error of the estimated feature position resulted from the neglecting of the distortion factor in the actual flight result. From Figure 39, it can be found that lens distortion impact on the X axis position error the most. At 100%, the estimated value is 50 meters closer than true value on average.

Secondly, lens distortion has more impact on features located far from the center of the image. Figure 40 shows feature position error vs projected feature distance from the optical center on image. Feature position error on X axis increases when feature lies further from the image center. On axis Y and Z, there is no obvious correlation between position error and feature distance from image center.

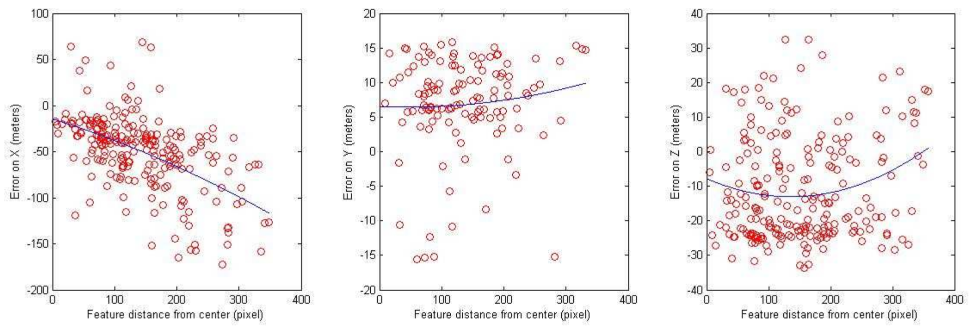
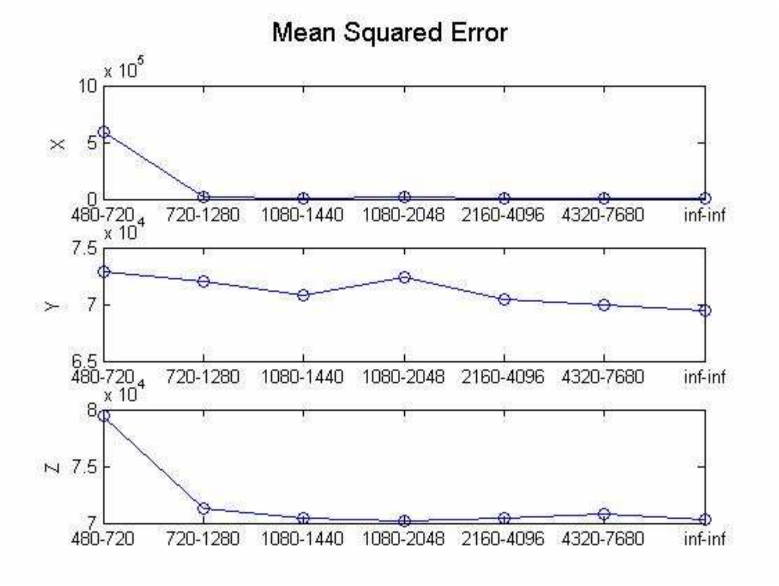


Figure 40 Estimated feature position error vs. feature distance on from image center on image plane.



7 Chapter: Future work

SLAM is not appropriate in dealing with moving obstacle in the scene, it is important to detect moving features and exclude it from the SLAM measurement list.

Appendices

[For the Appendices section title use Heading 1]

Appendix A Linearization of EKF

A.1 Equation for Initialization

A.2 Equations for Measurement Model

A.3 Equations for Composition Step

Appendix B Coordinate Transformation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{R_{base}} = Q \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}_{R_{mobile}}$$

where Q is the transformation matrix composed by rotation matrix on X, Y, and Z axis and a translation matrix. Q follows the TR?Y convention…(fig)

$$Q(r_x, r_y, r_z, T) = Q_{Rz}(r_z) \cdot Q_{Ry}(r_y) \cdot Q_{Rx}(r_x) \cdot Q_T(T)$$

$$Q^{-1}(r_x, r_y, r_z, T) = Q_T^{-1}(T) \cdot Q_{Rx}^{-1}(r_x) \cdot Q_{Ry}^{-1}(r_y) \cdot Q_{Rz}^{-1}(r_z)$$

$$Q_{Rx}(r_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(r_x) & -\sin(r_x) & 0 \\ 0 & \sin(r_x) & \cos(r_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{Ry}(r_y) = \begin{bmatrix} \cos(r_y) & 0 & \sin(r_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(r_y) & 0 & \cos(r_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{Rz}(r_z) = \begin{bmatrix} \cos(r_z) & -\sin(r_z) & 0 & 0 \\ \sin(r_z) & \cos(r_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_{Rx}^{-1} = Q_{Rx}^{T}$$

$$Q_{Ry}^{-1} = Q_{Ry}^{T}$$

$$Q_{Rz}^{-1} = Q_{Rz}^{T}$$

$Q_T^{-1} = Q_T$?

Appendix C

This is Appendix B.

C.1 Sub-Appendix

This is Appendix B, Sub-Appendix 1.

Bibliography

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

[2] J. A. Castellanos, J. Neira and J. D. Tardos, "LIMITS TO THE CONSISTENCY OF EKF-BASED SLAM".

[3] 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, October 2007.

[4] J. Civera, A. J. Davison and J. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics,* vol. 24, pp. 932-945, Oct. 2008.

[5] J. Civera, Ó. G. Grasa, A. J. Davison and J. M. M. Montiel, "1-Point RANSAC for EKF Filtering: Application to Real-Time Structure from Motion and Visual Odometry," *Journal of Field Robotics,* vol. 27(5), pp. 609-631, Oct. 2010.