Chapter 1

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 essential for the vehicle to navigate and avoid obstacle 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, 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 an obstacle 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. To achieve 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) [1]. 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.

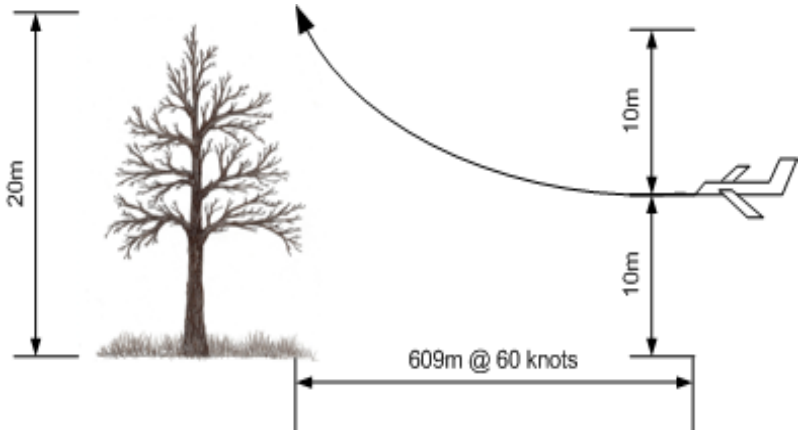

Figure 1: Case study for obstacle detection requirement

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 and . Error analysis compared to ground truth data is presented in and . The effect of using multiple sensors in improving the robustness is discussed in . At last, the camera baseline optimization is given in .

Chapter 2

REVIEW

2.1 SENSORS FOR OBSTACLE DETECTION

2.1.1 Overview

Many sensors can be used for obstacle detection such as ultrasonic sensor, laser range finder, sonar, or image sensor, 3D flash LIDAR [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12]. Ultrasonic sensors such as radar and sonar, and laser range finder are capable of providing high accuracy range measurement up to centimeter level, but only provides 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. Although LIDAR is becoming more and more popular on land robot platform, it is usually expensive [13] or have high power requirement and mass when long range is required [14]. 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 [12]. 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 [15] [16] [17] [18] [19] [20] [21] [8] [22] [23] [24].

2.1.2 Monocular Vision and Binocular Vision

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

For binocular camera setups, two cameras are placed apart from each other with their relative geometry (baseline separation) 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. Because binocular setup acquires two images simultaneously, the feature distance can be obtained at time 0. The challenge in binocular range sensing is feature correspondence between the two images. In addition, because of the limited distance in separation between the camera, and the vibration noise caused by vehicle motion, binocular range sensing are usually applied in short to medium range measurement.

For monocular camera setup, only one camera is used with an odometry sensors providing camera displacement measurement between frames. Then similar principle applies to triangulate the common scene point in two frames to measurement the distance of the point. If camera is facing the direction of travel, data association

is less difficult as image difference from frame to frame is usually small. Secondly, the baseline separation can be selected according to the targeted object distance by processing every frame, every other frames, or every n frames captured which enable monocular camera being used for longer range measurement. On the other hand, since monocular camera configuration capture 1 frames at a time, the earliest range measurement is available until two frames of images has been captured.

2.1.3 Camera Calibration

Camera model is what relating measurement on image plane to measurement of the real world. Without it, measurement of the 3D world through camera will be incorrect by a scale factor. Camera calibration gives a model of the camera's geometry and a distortion model of the lens. These models define the *intrinsic parameters* of a camera.

Basic Projection Geometry

For a 3D scene point $Q = (X, Y, Z)$ and the correspondent point $q = (x_{screen}, y_{screen})$ on the image plane, they are related by equation [25]

$$x_{screen} = f_x \left(\frac{X}{Z}\right) + c_x, \quad y_{screen} = f_y \left(\frac{Y}{Z}\right) + c_y \tag{1}$$

where (c_x, c_y) is the image plane coordinate at which optical axis intersects the image plane, f_x and f_y is the product of focal length f (unit millimeter) with the scaling factor s_x and s_y (unit pixel/millimeter) for X and Y. Therefore f_x and f_y is in unit pixel. The parameter c_x, c_y, f_x, f_y define the camera intrinsics matrix [25] [26].

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Lens Distortion

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

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

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

Tangential distortion arise from the misalignment between the image sensor and the lens. It is minimally characterized by two additional parameters p_1 and p_2

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

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

Camera Calibration Algorithm

Camera calibration is a process of finding the intrinsic parameter of a camera. In addition, since the camera's reference frame related to the world frame is usually unknown, the calibration process also calculate the status of the camera frame referenced

to the world frame, which is referred as the extrinsic parameters of the camera. There are two popular camera calibration methods. Tsai [27] requires at least 5 points in 3D space with known coordinate, and their corresponded coordinate in 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 position of the points is difficult to obtained. Zhang [28] proposed a more flexible method that requires at least two different views of a coplanar target, which is usually a black and white checker boards. However, to get more reliable result, 10 view of the checkerboard with different translation and rotation from the camera is desired. The target's coordinate in world frame need not to be known in advance. Instead, algorithm requires the size of the square on the check boards. Zhang's method is implemented in OpenCV [25], and is used to calibrate the camera for this work.

2.1.4 Computer Vision

Corner Detector

Corners, by Harriss definition, are places in the image where the autocorrelation matrix of the second derivatives has two large eigenvalues.Second derivatives are useful because they do not respond to uniform gradients.* Th is definition has the further advantage that, when we consider only the eigenvalues of the autocorrelation matrix, we are considering quantities that are invariant also to rotation, which is important because objects that we are tracking might rotate as well as move. Harriss original definition involved taking the determinant of H(p), subtracting the trace of H(p) (with some weighting coefficient), and then comparing this difference to a predetermined threshold. It was later found by Shi and Tomasi [Shi94] that good corners resulted as long as the smaller of the two eigenvalues was greater than a minimum threshold. Shi and Tomasis method was not only sufficient but in many

cases gave more satisfactory results than Harriss method. (from opencv)

2.1.5 Limitation of Optical Sensor in Recursive Algorithm

- Error Accumulation over Iterations
  - Feature quality Decreases over Iterations

2.1.6 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

The Kalman filter [29]published by R. E. Kalman in 1960 is a very powerful recursive data processing algorithm for dynamic stochastic processes. The filter find extensive use in control and navigation application for its ability of estimating past, present and even future state. It is an attractive candidate for 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 variable of interest. Given a dynamic process that satisfy the assumptions that Kalman filter is based on, the filter is the optimal algorithm in minimzing the mean of squared error of the state variable. This section briefly summerized assumption and formation of Kalman filter that's described in detail in [30] [31] [32] [33] [34]. A more intuitive introduction can be found in chapter 1 of [35].

The Kalman filter has three assumptions. 1) The system model is linear. The linearity is deisred in that the system model is more esily manipulated with engineering tool. When nonlinearities do exist, the typical approach is to linearize system model at some nominal points. 2)The noise embedded in system control and measurement is white. This assumption implies that the noise value is not correlated in time, and has equal power in all frequency. 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 densities require endless number of orders of statistic to describe the shape fully. Hence, when the probability density function of a noise process is non-Gaussian, the Kalman filter that propagates the first and second order statistic only include some of the information of the PDF, instead of all, as would be the case with Gaussian noise.

Kalman Filter Models

The Kalman filter requires two models. The process model define a discrete-time controlled process by a linear stochastic difference equation. The $n \times n$ matrix A relates the state variables x_{k-1} in previous time step $k-1$ to the state variabl x_k in the current time step k. The matrix B relates the optional control input μ to the state variables x. Given measurement vecor z_k of size $1 \times m$, the measurement model relates the state variables to the measurements by matrix H of size $n \times m$. The random variable w and v represent the uncertainty or noise of the process model, and measurement. w and v are assumed to be unrelated to each other, and has Gaussian distribution with covariance Q and R.

$$\text{Process Model: } x_k = Ax_{k-1} + B\mu_{k-1} + w_{k-1} \tag{3}$$

$$\text{Measurement Model: } z_k = Hx_k + v_k \tag{4}$$

The Algorithm

The Kalman filter operates in prediction and correction cycle after being initialized 2, with the state vector estimate $\hat{x}_k^-, \hat{x}_k^+$ contains the variable of interest at time step k, and state covariance matrix P_k^-, P_k^+ representing the error covariance of the estimate. The superscript - indicate the estimate is a priori (or predicted) estimate, and + indicate the estimate is a posteriori (or corrected) estimate. The equations

Table 1: Kalman Filter Operation Equations

Prediction	$\hat{x}_k^- = A\hat{x}_{k-1}^+ + B\mu_{k-1}$	(5)
	$P_k^- = AP_{k-1}^+A^T + Q$	(6)
Correction	$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}$	(7)
	$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$	(8)
	$P_k^+ = (I - K_k H)P_k^-$	(9)

used for prediction and correction are listed in 1. In prediction, an estimate of the state variables are made based on the known knowledge of the process (the process model). Since there are always unknown factor not fully described by the process model, the error of the estimate almost always increase in the prediction. During correction, a series of calculation were carried out to correct the a priori estimate. First, the predicted measurement $H\hat{x}_k^-$ are compared to the new measurement z_k. Their difference $z_k - H\hat{x}_k^-$ is called the measurement innovation, or residual. Next, the amount of residual is weighted by the Kalman gain K, and added to $\hat{x}_k^-$ as correction. The Kalman gain is fomulated so that it minimize the a posteriori error covariance matrix P_k^+.

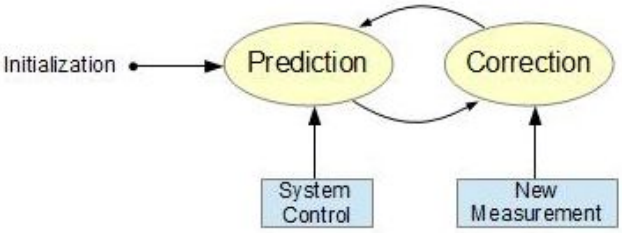

Figure 2: Kalman Operation Flow Diagram

Extended Kalman Filter

For a discrete-time controlled process, or its relationship with the measurements are non-linear, a Kalman filter must be linearized about the estimated trajectory, and

is referred to as an extended Kalman filter or EKF. A process with state vector x and measurement z that is governed by non-linear stochastic difference equation has process and measurement model

$$x_k = f(x_{k-1}, u_{k-1} + w_{k-1}), \tag{10}$$

$$z_k = h(x_k + v_k), \tag{11}$$

where the random varialbe w_k and v_k represent the process and measurement noise with variance Q and R. The the Kalman filter operation equations are given in table 2,

Table 2: Extended Kalman Filter Operation Equations

Prediction	$\hat{x}_k^- = f(\hat{x}_{k-1}^+, \mu_{k-1}, 0)$ (12)
	$P_k^- = A_k P_{k-1}^+ A_k^T + W_k Q_{k-1} W_k^T$ (13)
Correction	$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$ (14)
	$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$ (15)
	$P_k^+ = (I - K_k H) P_k^-$ (16)

where (subscript k omitted)

- A is the Jacobian matrix of partial derivatives of f with respect to x,

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

- W is the Jacobian matrix of partial derivatives of f with respect to w,

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

- H is the Jacobian matrix of partial derivatives of h with respect to x,

$$H_{[i,j]} = \frac{\partial h_i(\hat{x}_k^-, 0)}{\partial x_j}$$

- V is the Jacobian matrix of partial derivatives of h with respect to v,

$$V_{[i,j]} = \frac{\partial h_i(\hat{x}_k^-, 0)}{\partial v_j}$$

Note that when w and v directly describe the noise of state vector and measurement, the table 2 is the same as table 1.

Tuning

The tuning of the filter can be achieved by adjusting noise covariance matrix Q and R. The measurement noise covariance R is usually easy to estimate, ie., by taking some offline measurements to compute the variance. On the other hand, process noise covariance Q is more difficult. One common way is to inject enough uncertainty into the process noise, and only rely on reliable measurements.

2.2.2 SLAM with Extended Kalman Filter

The simultaneous localization and mapping(SLAM) problem answers to whether a robot can be make truly autonomous if it is plaed at an unknown location in an unknown environment. AFter many decades of research, the stucture of SLAM problem now has evolved to a standard Bayesian form. Two key approaches to solving

a SLAM problem is to use the extended Kalman filter (EKF-SLAM) and through the use of Rao-Blackwellized particle filter (FastSLAM). A thourough tutorial to the SLAM problem is given in [36] [37].

General EKF Model for SLAM

There is a high degree of correlation betwee the estimates of the location of the landmarks in map [38] [39]. This correlation exists because of the common error in the estimated vehicle location [40]. The implication of these works is that a consistent full solution to the SLAM problem require a joint state composed of the vehicle pose and every landmark position [36]. When the vehicle pose and landmarks position are fomulated as one single estimation problem, the result was convergent. The correlation between landmark play an important role in the quality of the solution. The more these correlation grew, the better the solution. [41] [42] [43] [44]

At a given time step k, the following variables are defined

- x_k describe the vehicle position and orientation.
- u_k the control vector applied at time $k-1$ to drive the vehicle to x_k at time k
- p_i a vector desecribe the location of the ith landmark. All landmarks locations are assumed to be time invariant.
- z_{ik} observation of the location of the ith landmark taken from the vehicle's location at time k

Then a complete state vector contains

$$\begin{bmatrix} \hat{x}_k \\ \\ \hat{p}_k \end{bmatrix}$$

and the state covariance matrix contains

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

The prediction and correction procedure can then be carried out following the standard EKF formulation. The lamdmarks are generally not updated at the prediction (ie. only x_k and P_{xx} are updated) unless they are moving.

Properties of SLAM

Dissanayake [44] reached three important convergency properties of Kalman filter based 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;*

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 result shows that the complete knowledge of cross correlation between landmark estimates is critical in maintaining the structure of a SLAM problem. The error in estimates of landmarks becomes more and more correlated as the vehicle venture throuth the unknown environment. The error eventually becomes fully correlated which means given the location of one landmark, the location of any other landmark can be determined with absolute certainty. As the map converges, the error in the

estimates of landmarks reduce to a lower bound determined by the error when the first observation was made.

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

Linearization Error and Consistency

Many researches reported and analyzed filter divergence due to linearization error [45], [46], [47], [48], [49]. As defined in [50], 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.

Huang investigated [51] on properties and consistency of nonlinear two-dimensional EKF based SLAM problem, derived and proved a number of theorem applied to EKF SLAM convergence and consistency issue, and concluded:

- Most of the convergence properties in [44] 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 (over optimistic estimation) are due to

    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.

SLAM for Building Large Scale Maps

Robot Centric Coordinate System In response to the consistency problem of classic EKF SLAM solution, some research adopted the *robotcentric mapping* [49] [52]. This approach uses a reference frame attached to the robot as the base frame. All geometric measurements and estimations are referred to the robot frame. This method was shown to improve the consistency of EKF based solution to SLAM, but using local map joining to produce a large map gives better result.

Submap Methods The submap methods are another mean to tackle large scale map problem. The submaps can be either globally referenced or relatively referenced. Global submaps estimate the location of the submap referenced to a common base frame [53] [54]. However, as the submap frames are referred to a common base frame, global submap does not improve the consistency issue caused by the robot pose uncertainty. [37]. Relatively referenced submap record its location referenced to the neighboring submaps [7] [6]. Global map is then obtained through vector summation. The relative submap is able to alleviate linearization problem in large scale map, but does not converge on global level without an independent submap structure. This problem can be solved by using Atlas framework and network coupled feature maps [55] [56]

2.3 Inverse Depth Parameterization for Distance Object

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 $\rho = 1/d$. 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 for initializing features into the EKF framework before it is safely triangulated. The inverse depth parameterization used in this work was first introduced in [10].

Chapter 3

Algorithm Description

The algorithm described in this chapter and its preliminary test result on real flight data was published in [18]. The program was implemented in Python programming language [57]. An open source machine vision library OpenCV was utilized to perform feature extraction and tracking. The feature extraction method used was the Shi-Tomasi corner detector [58]. Feature tracking was accomplished through the pyramid implementation of Lucas-Kanade optical flow method [59]. Both algorithms were implemented in OpenCV.

3.1 Camera Centric Inverse Depth Parameterization

The inverse depth parametrization by Civera [10] and camera centric coordinate system was adopted in this work with some modification to integrate the inertial measurements. Figure 3 shows the parameters definition in inverse depth parameterization.

A 3D scene point p_i^C can be defined by 6 parameters, with the superscript C

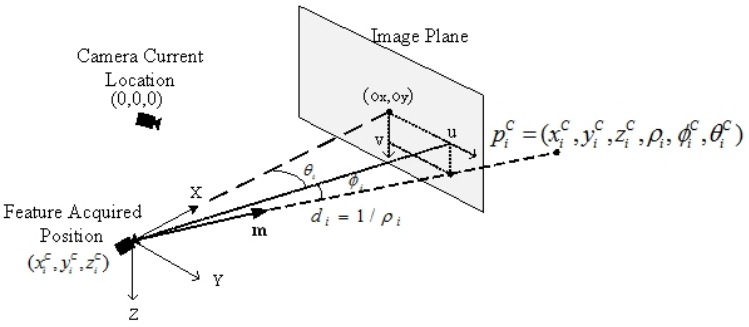

Figure 3: Inverse Depth Parameterization

representing a camera reference frame and i representing the feature number.:

$$p_i^C = \begin{bmatrix} x_i^C & y_i^C & z_i^C & \rho_i & \varphi_i^C & \theta_i^C \end{bmatrix} \tag{17}$$

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

$$m(\phi_i^C, \theta_i^C) = \begin{bmatrix} \cos\varphi_i^C \cos\theta_i^C \\ \cos\varphi_i^C \sin\theta_i^C \\ \sin\varphi_i^C \end{bmatrix} \tag{18}$$

Given the unit vector $[h_x, h_y, h_z]$, one can also find $[\varphi, \theta]$ by

$$\varphi = arctan\left(\frac{h_z}{\sqrt{h_x^2 + h_y^2}}\right) \tag{19}$$

$$\theta = arctan\left(\frac{h_y}{h_x}\right) \tag{20}$$

3.2 Modeling the System with Extended Kalman Filter

3.2.1 Full State Vector

The EKF state vector is defined as

$$x = \begin{bmatrix} OX_W^C & c^C & r^C & p_1^C & p_2^C & \dots \end{bmatrix}^T \tag{21}$$

where $OX_W^C = \begin{bmatrix} O_x^C & O_y^C & O_z^C & W_x^C & W_y^C & W_z^C \end{bmatrix}^T$ contains translation parameters $O_{x,y,z}^C$ and rotation parameters $W_{x,y,z}^C$ to transform the camera reference frame back to the world reference frame. $\left(c^C, r^C\right)^T$ 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

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 w^C in roll, pitch, and azimuth. The camera motion parameters at time k are then

$$\begin{bmatrix} OX_W^C & c^C & r^C & p_1^C & p_2^C & \vdots & p_n^C \end{bmatrix}_k^T$$
$$= \begin{bmatrix} OX_{W,k-1}^C & c_{measured}^C & r_{measured}^C & p_{1,k-1}^C & p_{2,k-1}^C & \vdots & p_{n,k-1}^C \end{bmatrix}^T \tag{22}$$

where

$$c^C_{measured} = v^C_{measured}\Delta t + \frac{1}{2}a^C_{measured}\Delta t^2$$

$$r^C_{measured} = r^C_{k-1} + w^C_{measured}$$

and Δt is the time elapsed from frame to frame.

3.2.3 Measurement Model

Each observed feature is related to the camera motion through the measurement model (figure 4). This relationship enables a correction on the camera motion and features parameters based on the features locations observed in the image.

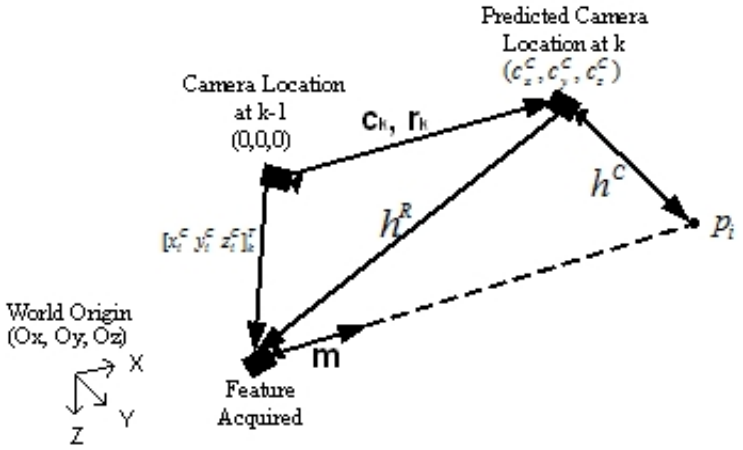

Figure 4: Measurement Model

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

$$h^R_k = \begin{bmatrix} x^C_i \\ y^C_i \\ z^C_i \end{bmatrix}_k - \begin{bmatrix} c^C_x \\ c^C_y \\ c^C_z \end{bmatrix}_k \tag{23}$$

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

$$h_k^C = Q^{-1}\left(r_k^C\right)\left(\rho_k h_k^R + m\left(\varphi_k^C, \theta_k^C\right)\right) \tag{24}$$

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

$$h_k^U = \begin{bmatrix} u_k \\ \\ v_k \end{bmatrix} = \begin{bmatrix} \frac{f_x h_{y,k}^C}{h_{x,k}^C} \\ \\ \frac{f_y h_{z,k}^C}{h_{x,k}^C} \end{bmatrix} \tag{25}$$

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. Detail formulation for linearization is given in appendix **??**

3.2.4 Composition Step

The correction using measurement model described above has its camera frame at step $k-1$ as its reference frame. Therefore, a coordinate transformation must be done to transform all parameters, and error matrix into camera frame at step k so that the next cycle of tracking can be continued. World reference frame parameters from step $k-1$ to step k is related by

$$\begin{bmatrix} O_x^{C_k} \\ O_y^{C_k} \\ O_z^{C_k} \end{bmatrix}_k = Q^{-1}(r_k^{C_{k-1}}) \left(\begin{bmatrix} O_x^{C_{k-1}} \\ O_y^{C_{k-1}} \\ O_z^{C_{k-1}} \end{bmatrix}_k - \begin{bmatrix} c_x^{C_{k-1}} \\ c_y^{C_{k-1}} \\ c_z^{C_{k-1}} \end{bmatrix}_k \right) \tag{26}$$

$$\begin{bmatrix} W_x^{C_k} \\ W_y^{C_k} \\ W_z^{C_k} \end{bmatrix}_{k-1} = \begin{bmatrix} W_x^{C_{k-1}} \\ W_y^{C_{k-1}} \\ W_z^{C_{k-1}} \end{bmatrix}_k - r^{C_{k-1}} \tag{27}$$

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

$$\begin{bmatrix} x_i^{C_k} \\ y_i^{C_k} \\ z_i^{C_k} \end{bmatrix}_k = Q^{-1}(r^{C_{k-1}}) \left(\begin{bmatrix} x_i^{C_{k-1}} \\ y_i^{C_{k-1}} \\ z_i^{C_{k-1}} \end{bmatrix}_k - \begin{bmatrix} c_x^{C_{k-1}} \\ c_y^{C_{k-1}} \\ c_z^{C_{k-1}} \end{bmatrix}_k \right) \tag{28}$$

$$\begin{bmatrix} \rho_i \\ \varphi_i^{C_k} \\ \theta_i^{C_k} \end{bmatrix}_k = \begin{bmatrix} \rho_i \\ m^{-1}\left(Q^{-1}(r^{C_{k-1}}) m(\varphi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}}) \right) \end{bmatrix} \tag{29}$$

where ρ_i is a scaler, and doesn't need transformation. $m(\varphi_{i,k}^{C_{k-1}}, \theta_{i,k}^{C_{k-1}})$ is the unit vector pointing from the initialization point to the feature seen by the camera at step $k-1$. m^{-1} represent the operator that convert the unit vector back to $[\varphi, \theta]$ angles as defined in equation (19) (20).

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

$$P_k^{C_k} = J_{C_{k-1}\to C_k} P_k^{C_{k-1}} J_{C_{k-1}\to C_k}^T \tag{30}$$

where $J_{C_{k-1}\to C_k}$ is the Jacobian matrix of the composition equations. Detail derivation of the Jacobian matrix is given in appendix **??**

3.2.5 Filter Initialization

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,

$$OX_W^C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{31}$$

$$c^C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{32}$$

$$r^C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{33}$$

$$p_i^C = \begin{bmatrix} 0 & 0 & 0 & \rho_i & \varphi_i^C & \theta_i^C \end{bmatrix}^T \tag{34}$$

The inverse distance ρ of all features are initialized to 0.1 ($d = 100m$). The features elevation-azimuth pair $[\varphi_i^C, \theta_i^C]$ is extracted from features coordinates in image plane. First, a vector pointing from camera optical center to feature can be defined by

$$h^C = \begin{bmatrix} h_x^C \\ h_y^C \\ h_z^C \end{bmatrix} = \begin{bmatrix} 1 \\ (u - c_x) \cdot f_x \\ (v - c_y) \cdot f_y \end{bmatrix} \tag{35}$$

where $[u, v]$ is the feature coordinate in the image, $[f_x, f_y]$ is the scaling factor of the projection from the scene to image plane, $[c_x, c_y]$ is the coordinate where the optical axis intersects the image plane. The elevation-azimuth pair $[\varphi_i^C, \theta_i^C]$ can then be directly calculated from h^C using equation (19) and (20)

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 frame coordinate and orientation, and the camera motion is

$$P = I_{12\times 12} \cdot \epsilon \tag{36}$$

where I is a 12×12 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

$$P_{new} = J \begin{bmatrix} P_{old} & 0 \\ 0 & R \end{bmatrix} J^T \tag{37}$$

where P_{old} is the covariance matrix of the existing state vector, and the initial P_{old}

before the first feature is added is given in equation (36). Matrix R is given by.

$$R = \begin{bmatrix} \sigma_{x_i^C} & & & & & \\ & \sigma_{y_i^C} & & 0 & & \\ & & \sigma_{z_i^C} & & & \\ & & & \sigma_\rho & & \\ & 0 & & & \sigma_{image} & \\ & & & & & \sigma_{image} \end{bmatrix} = \begin{bmatrix} \epsilon & & & & & \\ & \epsilon & & 0 & & \\ & & \epsilon & & & \\ & & & 0.1 & & \\ & 0 & & & 1 & \\ & & & & & 1 \end{bmatrix} \tag{38}$$

where$[\sigma_{x_i}^C, \sigma_{y_i}^C, \sigma_{z_i}^C]$ is the uncertainty of the camera optical center position, which is the same as the world frame coordinate, initialized to ϵ. σ_{image} is the variance of feature coordinate on image plane, set to 1 pixel. $\sigma_\rho$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 in equation(37) is the Jacobian matrix for the feature initialization equations given by equation(31)-(35), (19), and (20). Detail formulation of the Jacobian matrix is given in appendix **??**.

3.2.6 Adding and Deleting Features

Features that move out of the camera's FOI are removed from the filter state vector and covariance matrix to keep filter size managable. The removal is done by deleting the parameters from the state vector, and the related rows and columns from the state covariance matrix. All tracked features are recorded into a database. The deleted ones still gets their parameters updated based on the camera motion, but no Kalman

corection is done.

To maintain sufficient amount of features for mapping, new features are acquired and added to the filter should a feature deletion occurred, or the number of tracked feature is lower than a threshold. The feature addition occurs after the composition step of an iteration, and follows the same procedure as the feature initialization described previously.

Chapter 4

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 5)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.

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 flight control INS/GPS navigation unit Athena GS-111m [60](Figure 6). Analog video and navigation data are sent to the helicopter via three BNC cables and one data cable. Installed in the

Figure 5: Simulated UAS towed by helicopter

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 7). 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 8.

Figure 6: Sensors mounted on SUAS. Top: the SUAS, bottom left: Athena GS-111m, bottom right: GPS antenna

Figure 7: Compact PCI data acquisition system (CDAC)

Figure 8: Image from monocular camera with GPS second timestamp

4.2 Camera Calibration

A camera calibration was performed after the flight by taking a piece of video of a checker board pattern with various translation and rotation motion. A total of 20 views of the calibration target was chosen from the video, and fed to the calibration algorithm. A few examples were shown in figure 9.

The algorithm,"calibration.exe", used for calibrating the camera comes with the OpenCV installation. The digitized images has a resolution of 720 pixels in width and 480 pixels in height. Table 3 below listed the calibration results.

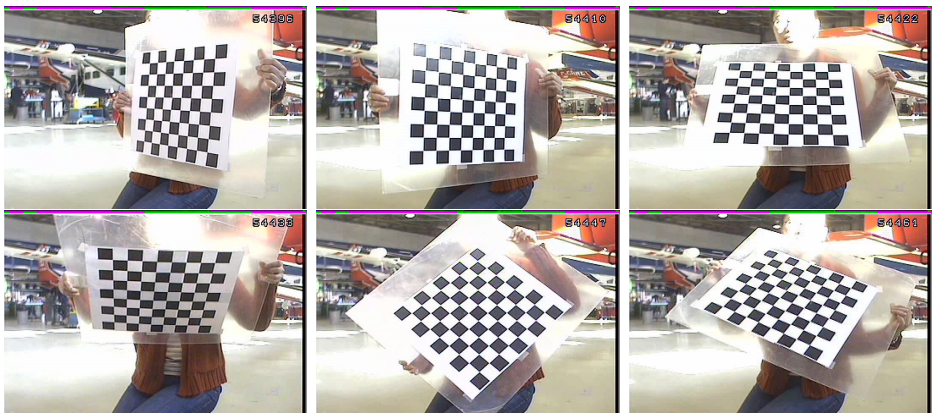

Figure 9: Camera Calibration Inputs

Table 3: Camera Calibration Result

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

4.3 Ground Truth Data Collection and Comparison

UAS localization ground truth are obtained through on board flight control unit GS-111m. The unit records the SUAS position in GPS longitude and latitude coordinate. Orientation is obtained from the roll pitch and heading measurements. Roll and pitch accuracy has 0.1° mean and 0.1° standard deviation. Heading accuracy can achieve 0.5°. [60]

The estimated SUAS position can be directly obtained from the inverse of world origin estimation in the CC_EKF_SLAM state vector: $[O^c_{XYZ}, W^c_{XYZ}]$.

Digital elevation map (DEM) are downloaded from CGIAR-CSI website [61] and used as ground truth data for feature mapping. The downloaded DEM contains longitude, latitude and see level elevation of the terrain with a resolution approximately 100 meters by 100 meters. To compare estimated feature position to the the DEM, some conversion are necessary to bring both data to the same coordinate system. In this work, the comparison are done in UTM coordinate.

First, the longitude and latitude in DEM data are converted into UTM using

the WGS84 world geodetic system [62]. Many library are readily available to do the conversion by taking GPS coordinate and zone number as input. The library used in this work is a python interface to PROJ.4 library [63] called pyproj [63]. Secondly, the DEM data in UTM were converted into world frame using transformation matrix with initial SUAS position and orientation as input.

To bring result from the CC_EKF_SLAM algorithm to world frame, feature coordinates must be first converted into world frame using the estimated UAS localization results. Let $[X_i^W, Y_i^W, Z_i^W]^T$ be the feature coordinate in world frame, it can be calculated by

$$\begin{bmatrix} X_i^W \\ Y_i^W \\ Z_i^W \end{bmatrix} = Q^{-1}(O_{XYZ}^c, W_{XYZ}^c) \left(\begin{bmatrix} x_i^C \\ y_i^C \\ z_i^C \end{bmatrix} + \frac{1}{\rho_i} m(\varphi_i^C, \theta_i^C) \right) \tag{39}$$

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 $[Northing_{init} \quad Easting_{init} \quad Height_{init}]$. Then, a transformation matrix can be defined as followed:

$$Q_{Roll}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(Roll_{init}) & \sin(Roll_{init}) & 0 \\ 0 & -sin(Roll_{init}) & \cos(Roll_{init}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$Q_{Pitch}^{-1} = \begin{bmatrix} \cos(Pitch_{init} + \pi) & 0 & -\sin(Pitcch_{init} + \pi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(Pitch_{init} + \pi) & 0 & \cos(Pitch_{init} + \pi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{41}$$

$$Q_{Azimuth}^{-1} = \begin{bmatrix} \cos(Azimuth_{init} + \frac{\pi}{2}) & \sin(Azimuth_{init} + \frac{\pi}{2}) & 0 & 0 \\ -sin(Azimuth_{init} + \frac{\pi}{2}) & \cos(Azimuth_{init} + \frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

$$Q_{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -Northing_{init} \\ 0 & 1 & 0 & -Easting_{init} \\ 0 & 0 & 1 & -Height_{init} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{43}$$

$$Q^{-1} = Q_{T}^{-1} \cdot Q_{Roll}^{-1} \cdot Q_{Pitch}^{-1} \cdot Q_{Azimuth}^{-1} \tag{44}$$

At last, the DEM in world coordinate is given by

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

Chapter 5

Result From Flight Data

This chapter is divided into three main sections. Firstly, the algorithm's convergence and consistency was analyzed. Secondly, the accuracy of the algorithm is examined by comparing to ground truth data. The third section summarized test results for tuning the algorithm for better accuracy and efficiency. The forth section present the advantage of using IMU data. The fifth section outline the inadequacies of the CC_EKF_SLAM algorithm identified.

To test the performance of CC_EKF_SLAM algorithm, 4 segments of video were selected from the test flight video, and 400 frames were processed in each piece. The filter initialized 40 features at the first frame, and maintain the tracked features amount at this number by initializing new features when existing features moved out of FOV.

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 and Consistency

5.1.1 Convergence and Tracking

Among the feature parameters, the feature initialization point were initialized to the zero which is the origin of the camera centric coordinate. ϕ and θ were calculated directly from the feature position on image plane, have high accuracy and don't require convergence. The only parameter that goes through a converging process is the features' inverse depth ρ, which were initialized to 0.1 for all features. Figure 10 shows the $1/\rho$ plot for video segment1 over 200 frames. The depth estimators went through rapid changes for several frames after their initialization. Within approximate 20 frames, most estimtors settles to a stable value. The estimated features distance ranged from 400 meters to about 1500 meters, confirming the algorithm's capability for estimating features at great distance. On the other hands, some features take a long time to settle, while some other never settled, such as 12 and 20.

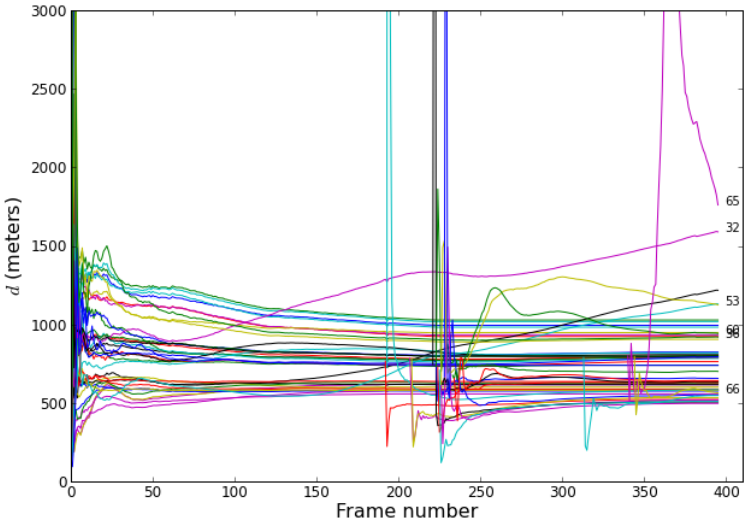

Figure 10: Inverse Depth Convergence

Although the features initialization point $[x_i, y_i, z_i]$, and the deviation-elevation angle pair $[\phi, \theta]$ did not go through a converging stage, they do get updated and

converted into the new camera coordinate using the estimated camera motion at every iteration. As a result, the accuracy of these parameters varies. Ideally, the coordinates should converge to a fixed value. However, the plot indicates a variation as the vehicle travels along. Figure 11 shows the tracking of these parameters over the entired processed frames. These parameters were stable for about 200 frames. As soon as any feature was deleted from the filter and new features added, the parameters of the feature after being converted into world frame start to drift. In 11 the deletion of old feature is marked by a vertical gray dash line. It shows that the parameters of the deleted feature start to drift right after that line. The y_i and z_i are most affected. The behavior is caused by the estimated error in the SUAS localization. In order to compensate the error in localization(Figure 13), the algorithm made adjustment on the estimate of features parameters so that the combinational result agrees with the measurement. On the other hand, features mapping is not affected too much by the error in localization. Because their parameters are transformed in each iteration to the new camera frame using the estimated SUAS motion which carries the error. As long as their parameters are updated together with the estimated motion on that iteration, and transformed back to the world frame using the same estimated motion, the final result of features location in world frame is inaffected by the error in the SUAS localization. However, for feature removed from the filter, their parameters are no longer updated, therefore revealing the error in SUAS localization.

5.1.2 Consistency Analysis

A EKF system becomes inconsistent when the variance of state vector element becomes too small and forbid an effective update. In addition, the uncertainty of the SUAS position should increases as it move away from the origin. To examine the consistency of the CC_EKF_SLAM algorithm, the variance of all state vectors for all procesed frames were extracted and plotted in figure 12. The two plots on the left

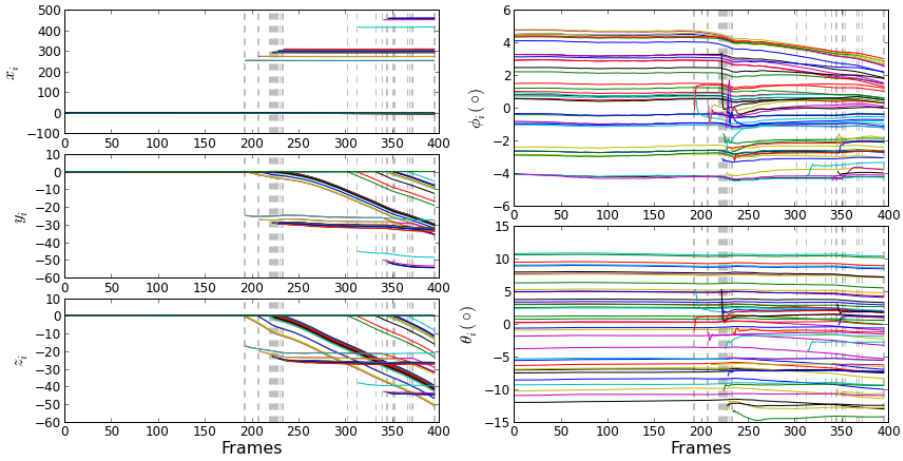

Figure 11: Feature parameters tracking

shows the variance of world frame position and orientation in camera frame. The three plots on the right show the variance of feature parameters. The variance of world frame position increases with iterations. On the other hand, world frame orientation decreases with iterations. Especially the orientation in Y and Z axis remain below 5e-7 for the most time. For feature parameters, all variance decreases with iterations, and their value is very small.

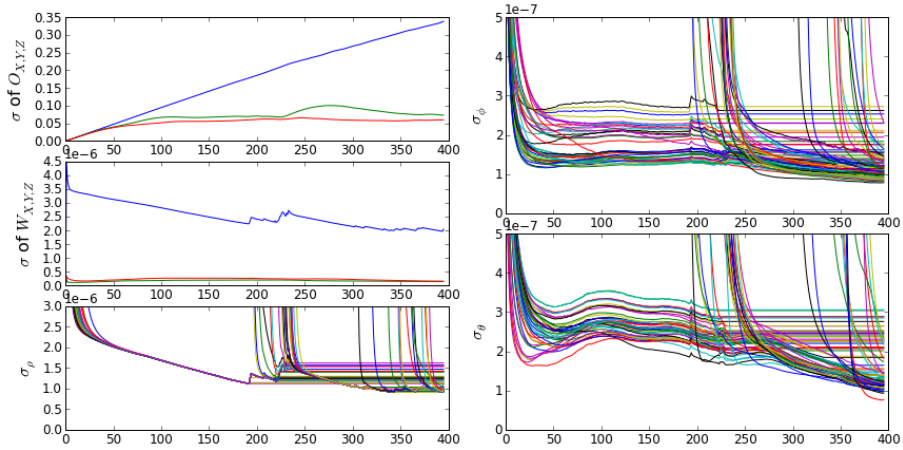

Figure 12: State vector variance

To see how variance affect the correction the filtered made on the state vector, the correction applied each iteration were plotted ni figure 13. The 1^{st} column shows

the correction on world frame position in camera frame. The 2^{nd} column shows the correction on feature initialization coordinate in camera frame. The 3^{rd} column shows the correction on features parameters ρ, ϕ and θ. It can be observed that world frame position on Y and Z component receive more correction than the X component, despite their variance is smaller than the X axis. Similarly, feature parameter θ and ϕ receive more correction than ρ despite their variance is smaller than the variance of ρ. Secondly, there is a strong inverse correlation between the Y and Z component of feature initialization position and the world frame position on Y and Z axis. Similarly, ϕ is correlated to world frame position Z component, θ is correlated to the world frame position Y component. Since the features initialization coordinate has high correlation with the world frame Y and Z position, the main contributor all the drift comes from the world frame position correction. Figure 14 shows the Kalman Gain over the 5 iteration for the first 35 elements in the state vector, which include world frame parameters, SUAS motion parameters, and parameters for the first 4 features. At 1^{st} iteration, the high gain is on the feature parameter ρ. Starting from the 2^{nd} iteration, more and more weight goes into the Y and Z component of world frame position, as well as the Y and Z component of features initialization position. Although the variance of these two parameters is small, it is possible that the linearization of the measurement model increased the correction gain on these parameter. This issue requires further digging, and will be analyze as a future work.

In summary of the consistency analysis, the filter has not moved into inconsistent state at 400 iterations. However, the variance of parameters does decrease at each iteration that it is only a matter of time that the filter will become inconsistent. Therefore, the filter will require a reset procedure to

1. Resync the SUAS location with GPS.

2. Reset its covariance matrix.

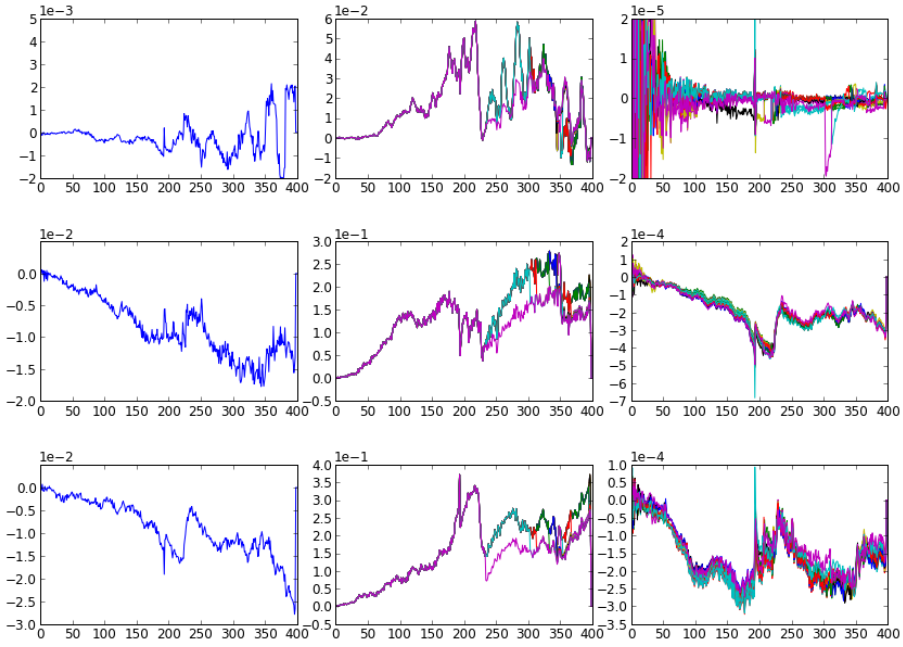

Figure 13: State Vector Corrections

for any large area operation.

5.2 Accuracy

5.2.1 SUAS Localization

The SUAS location and orientation were verified in UTM coordinate. Ground truth data comes from the GPS for position, and magnetometer for orientation. The GPS positioning can generally achieve 7.8 meters in accuracy [64]. For orientation, accuracy on the datasheet specified 0.1° for roll and pitch, and 0.5° for heading. Figure 15 shows the estimated SUAS position and orientation, ground truth data, and the error. From the comparison, X position error reached 20m maximum, while Y and Z position error is bigger, reaching 50 meter and 30 meters respectly. For orientation, the estimated value agrees with the ground truth pattern, with maximum error within 0.02 rad or 1.15°. Although the error of pitch is biased to the negative side, and error

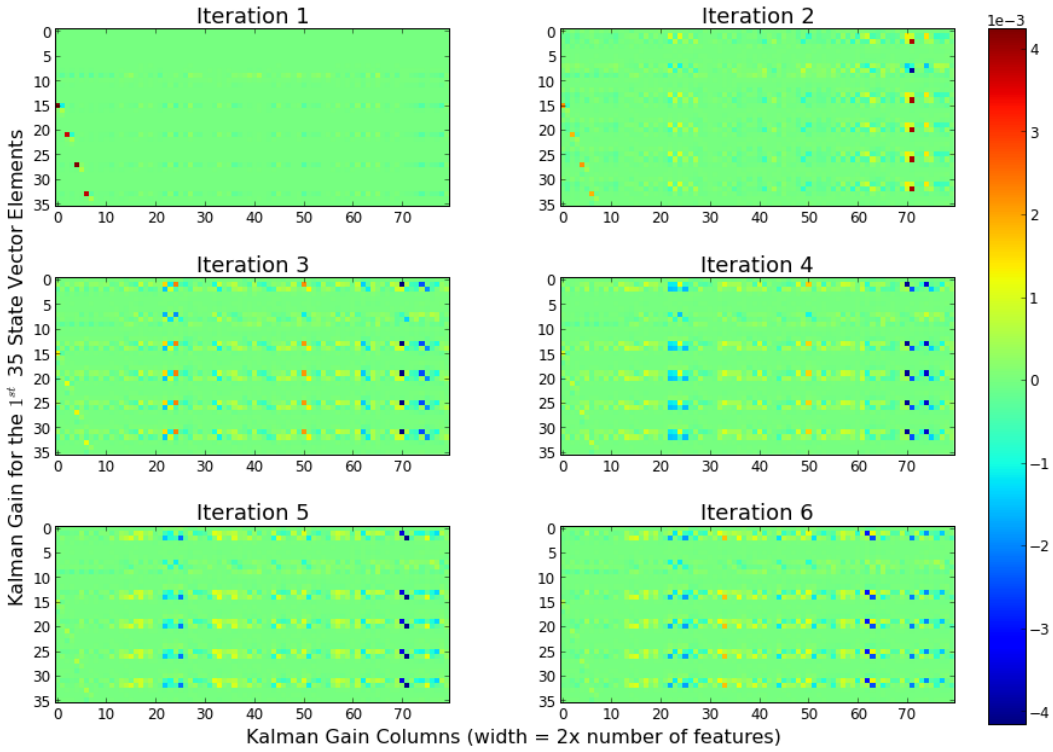

Figure 14: Kalman Gain Matrix for the first 35 State Vector Elements

of heading is biased to the positive side. there was not sign of error diverging within the 400 frames processed.

5.2.2 Features Mapping

Although raw ground truth data for the terrain was downloaded from [61] CGIAR-CSI website, making data correspondence with the estimated feature is non-trival for natual scene where it lacks of signature landmark sucn as building corners. From the experience of analyzing simulation data (chapter 6), the initialized feature deviation and elevation angle ϕ and θ agrees very well to the ground truth value. In addition, feature initialization position is set to $[0, 0, 0]$ in camera frame, therefore carris no error. Therefore, the following procedure are used the find the corresponding ground truth location for any estimated feature.

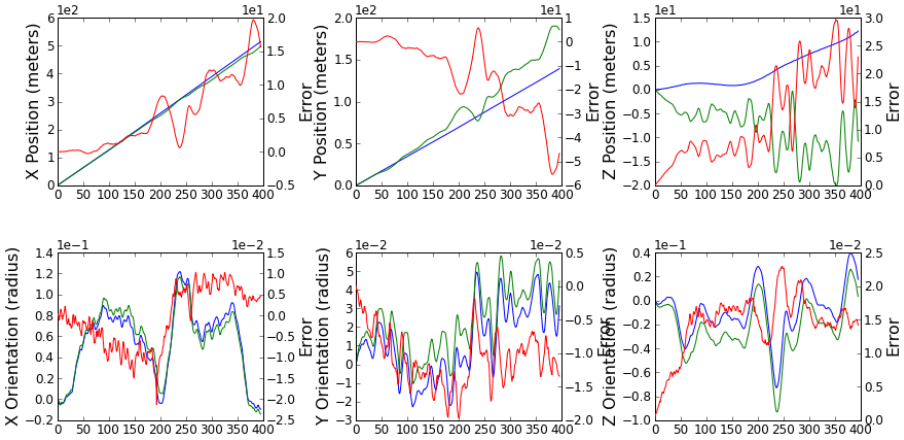

Figure 15: SUAS position and orientation

1. The frame number at which feature was initialized was identified, and ground truth location of the SUAS at that frame was recorded.

2. Shift DEM to the initialization point using the SUAS ground truth location recorded

3. Align the feature deviation θ and elevation ϕ angle to UTM frame. (These angles were recorded in camera frame)

4. Create a vertical plane using θ and slice the DEM to form a 1D elevation plot (Figure 16 blue line)

5. Create a line using ϕ and intersect the 1D elevation plot (Figure 16 green line). The 1^{st} intersection point is used as ground truth feature location.

By comparing to the ground truth feature location extracted using the method describe above, figure 17 shows the feature position error convergence plot in world frame. Feature locations are compared in world frame for the ease of corresponding the estimated features and terrain plot with the video by eye. The error convergence is plotted in figure 17 top left. X component (along which axis the SUAS is traveling) of feature position converge to near zero in general with maximum error less than

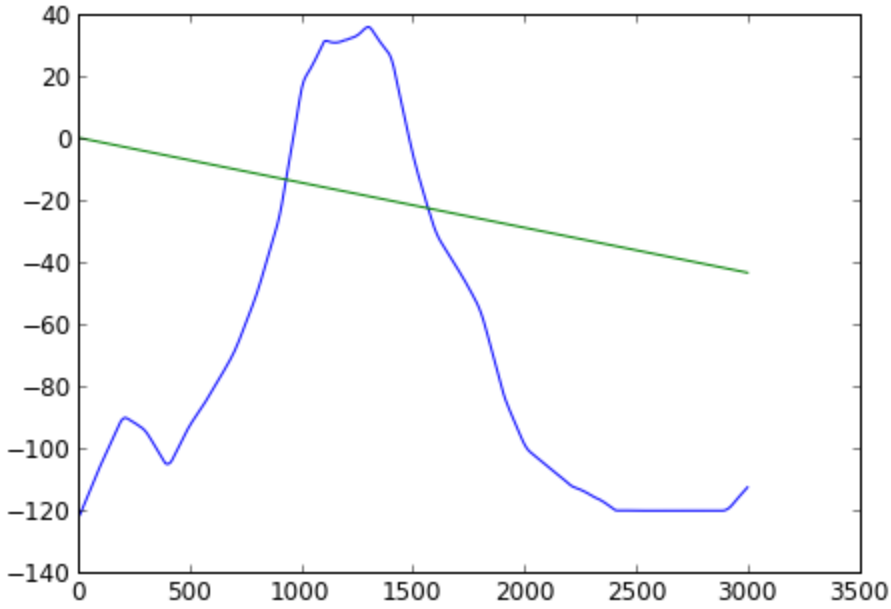

Figure 16: Feature position error

120m for the converged features. Y component shows a clear offset for features not initialized at first frame. This behavior is similar to what's seen in the simulation analysis 6.2.2 where offset in features position estimates are caused by drift in SUAS localization estimation. In this case, the features position were converted into world frame using the estimated SUAS position, while the ground truth feature position are found using the ground truth SUAS position, which is different than the estimated. Therefore the offset error can be seen. Z component does not show such offset, but some features' Z component position drift away after new features were added into the filter. This drift could be related to the drift in estimates of angle ϕ (figure 11).

Plotting the actual estimated and ground truth features position reveals relation between the error and the feature's position. Figure 18 bottom plots the feature positions and error at frame 398. First on X axis, the biggest error occur on feature number 65 which is initialized pretty late in the sequence, and has not converged properly. On Y axis, the position plot also shows feature number higher than 40 carries an offset error. In addition, the average error is increasing as feature number

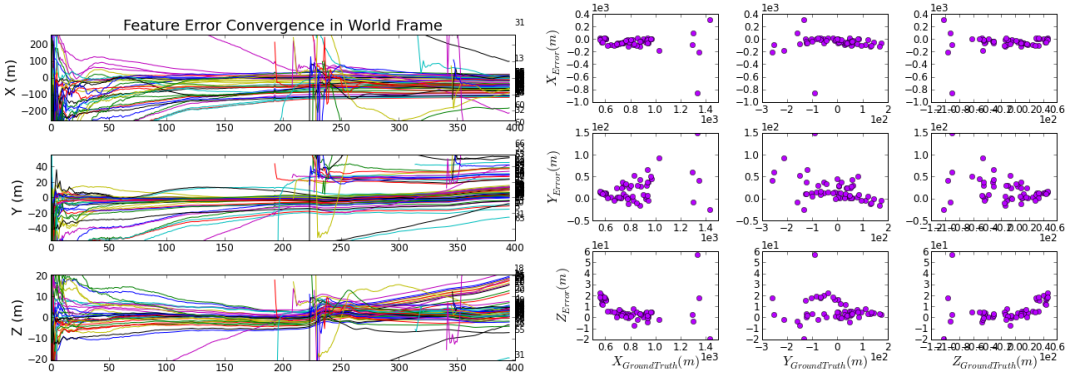

Figure 17: Left: , right:

gets bigger, which indicate the further the feature initialization point is from the world origin, the bigger this offset error will me. On Z axis, besids the feature 65 carries a big error, features with ground truth Z position bigger than 25 generally has an error of 15-20 meters. If the error are plotted against the ground truth position (figure 17 right), it can be seen that these features all located within 600 meters to the camera.

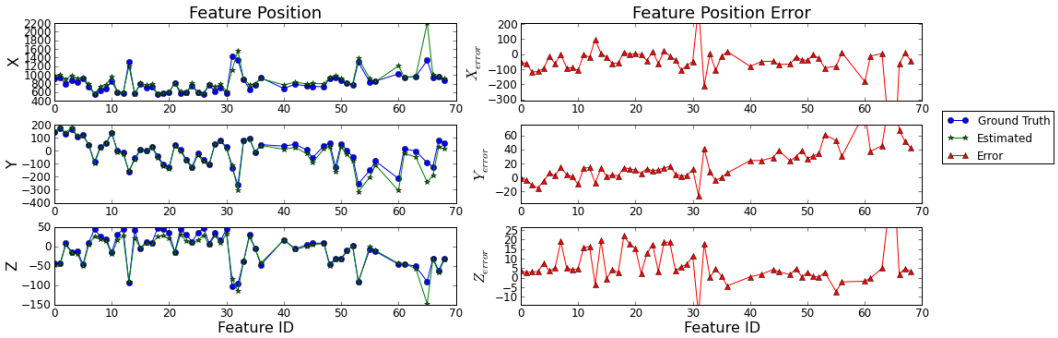

Figure 18: Feature position and error. Left: , right:

Using the estimated feature position, a terrain map can be generated. Figure 19 shows the terrain map generated from the feature position analyzed above. On the right, the ground truth DEM is also shown for comparison.

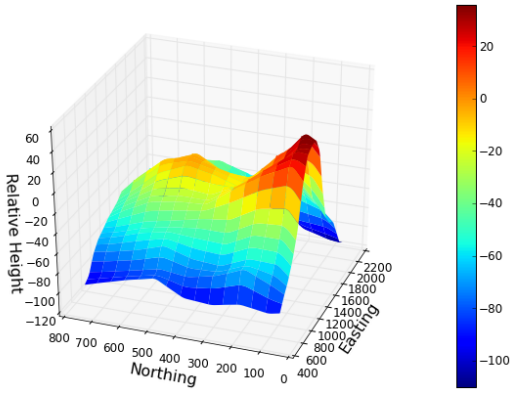

Figure 19: Terrain map. Left: estimated, right: ground truth

5.3 Compare to Visually Corresponded Feature

Because it is hard to correspond feature extracted from video to the actual data on DEM, a second piece of video was processed with man-made structure in the scene so that feature correspondence can be made manually. Figure 20 shows a zoomed in view of the features extracted by the CC_EKF_SLAM. All features are located at the bottom left corner of the image. Figure 21 shows the common features found manually on the satalite imaging on Google Earth [].

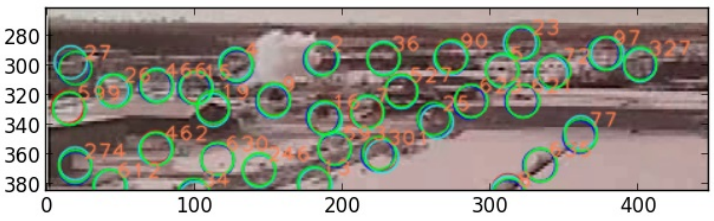

Figure 20: Feature identified by algorithm

Figure 21: Common features idetified manually on Google Earth

The SUAS position and orientation were compared to onboard GPS and INS recording, and the result are plotted in figure 22 and figure 23. Similar to the natual scene video, SUAS location is more accurate on X, and shows more drift on Y and Z coordinate. The drift become significant when frame number exceeds 200. However, with the SUAS decending, drift on Z is less than when it is ascending. For orientation error, roll and pitch error oscillate around zero, but azimuth error drift away as frame number increases.

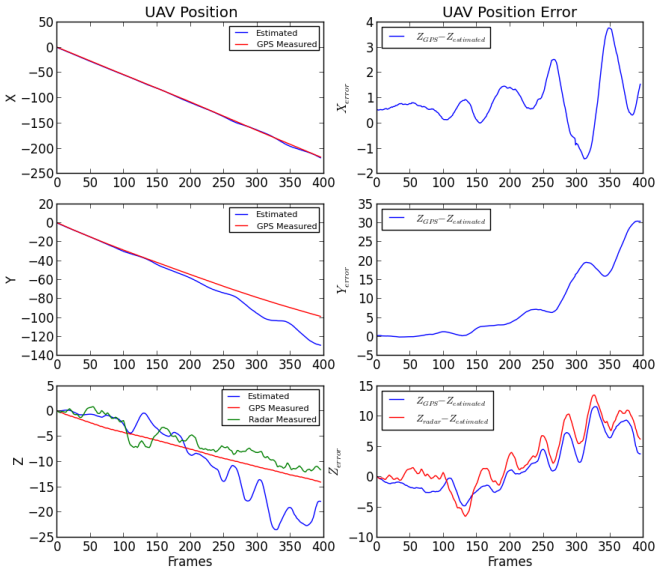

Figure 22: SUAS position accuracy compared to GPS

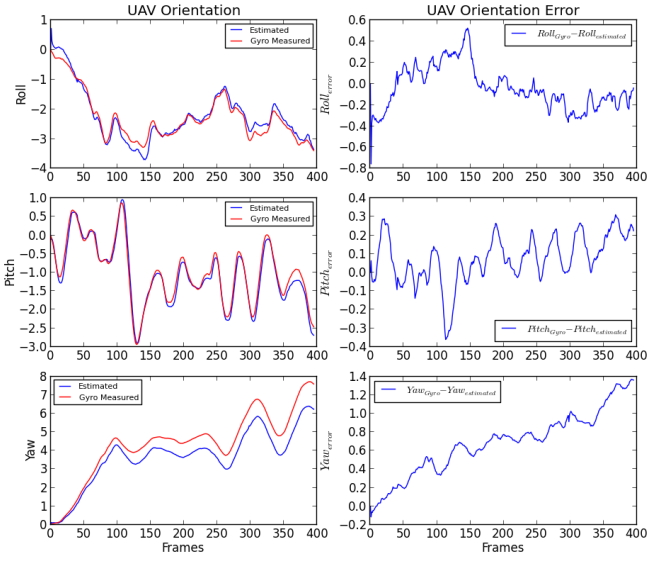

Figure 23: SUAS orientation accuracy compared to ??

Figure 24 shows the features location in X and Y axis. The google earth map provide a very rough elevation data, and was not used. The ground truth X and Y coordinate of the features were measured on Google Earth in GPS coordinates and converted into UTM. This plot shows a clear offset error between the ground truth and the estimated. X axis coordinate has a mean offset is about 100 meters, and Y axis coordinate has a offset of about 130m. Since all features are located in the bottom left corner of the FOV, it is possible that lens distortion plays a important part in contributing to these error.

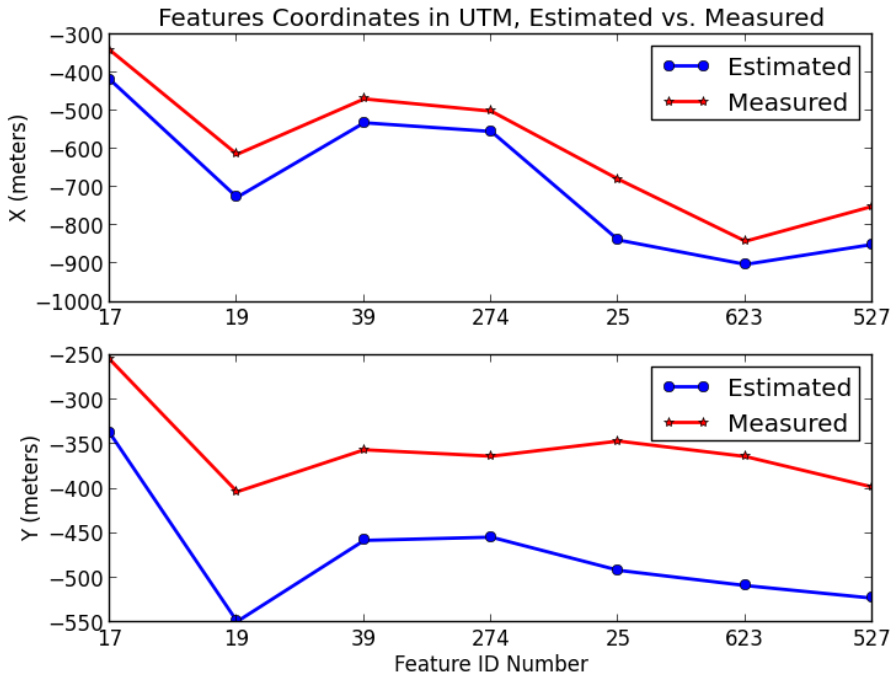

Figure 24: Feature error on X and Y axis.

Chapter 6

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 (Figure 25). 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.

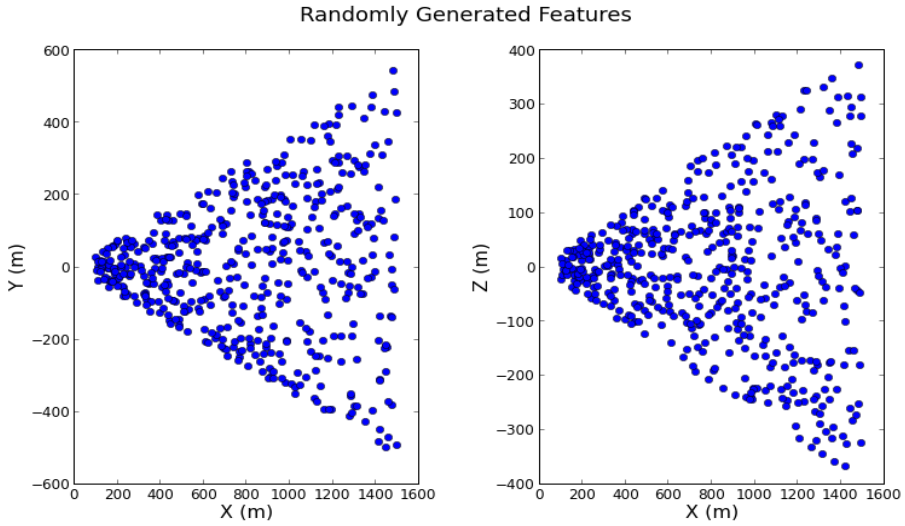

Figure 25: Randomly Generated 3D Feature Points

- 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 26 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 $Estimated - GroundTruth$ 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.

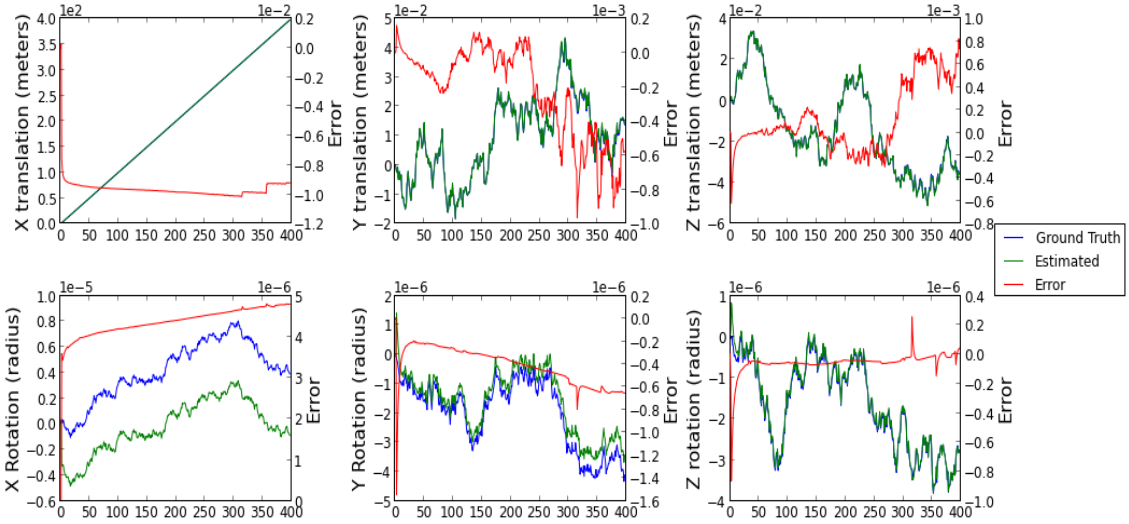

Figure 26: UAS localization error under no noise condition

6.1.2 Features Mapping - Convergence and Accuracy

Figure 27 top left shows feature parameters $[d, \varphi, \theta]$ (where $d = 1/\rho$) plotted against frame number for the first 50 frames. The feature depth d for all features converged within 3 frames; elevation-azimuth angles $[\varphi, \theta]$ stay almost constant after initialization. A more detail graph can be seen from the error convergence plot for these parameters (Figure 27 top right.) 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. $[\varphi, \theta]$ show small drift within $+/-0.0002°$ respectively. However, as tracking continue into later frames, error of $[\varphi, \theta]$ gradually grew bigger. The resulting error for feature coordinate in world frame represented in standard Euclidean XYZ parameterization is plotted in figure 27 bottom left and right. 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

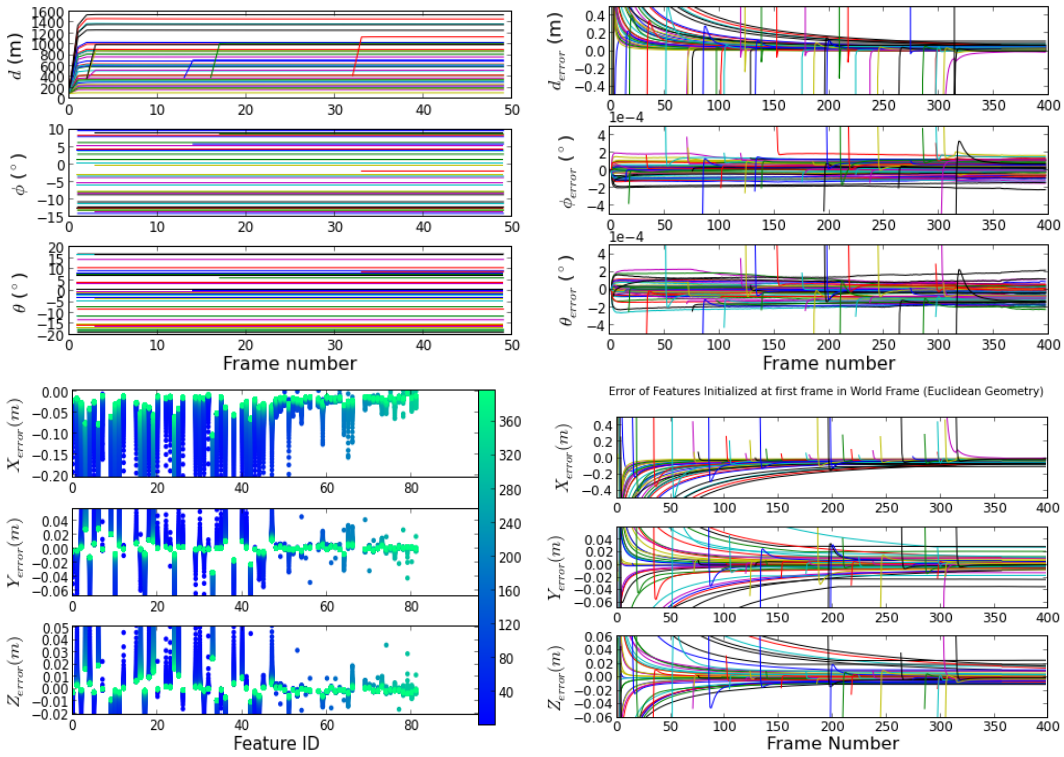

Figure 27: Features parameters and error convergence under no noise condition

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 28 shows the UAS localization error statistic under translation motion on Y and Z axis and 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.

To understand how rotation motion affect the UAS localization estimation, the UAS position on X, Y, Z in world frame are plotted below (figure 29) with rotation

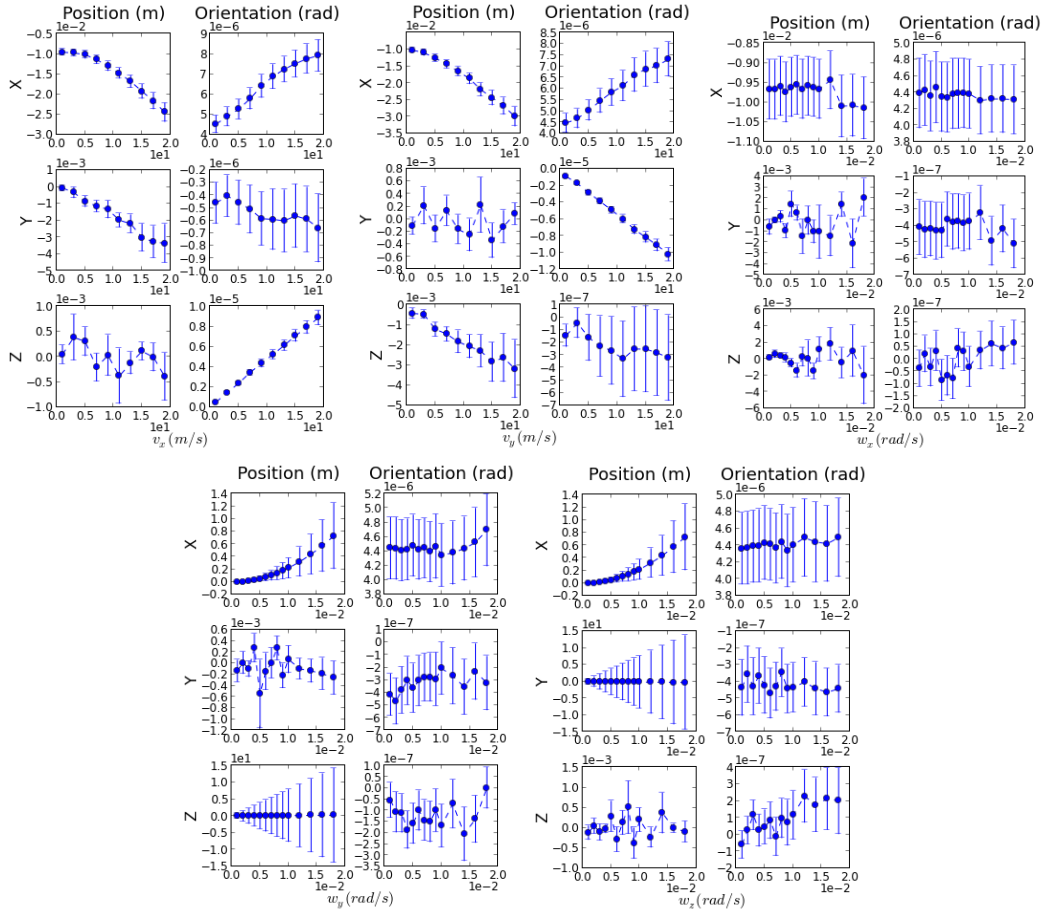

Figure 28: UAS localization error under translational motion

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.

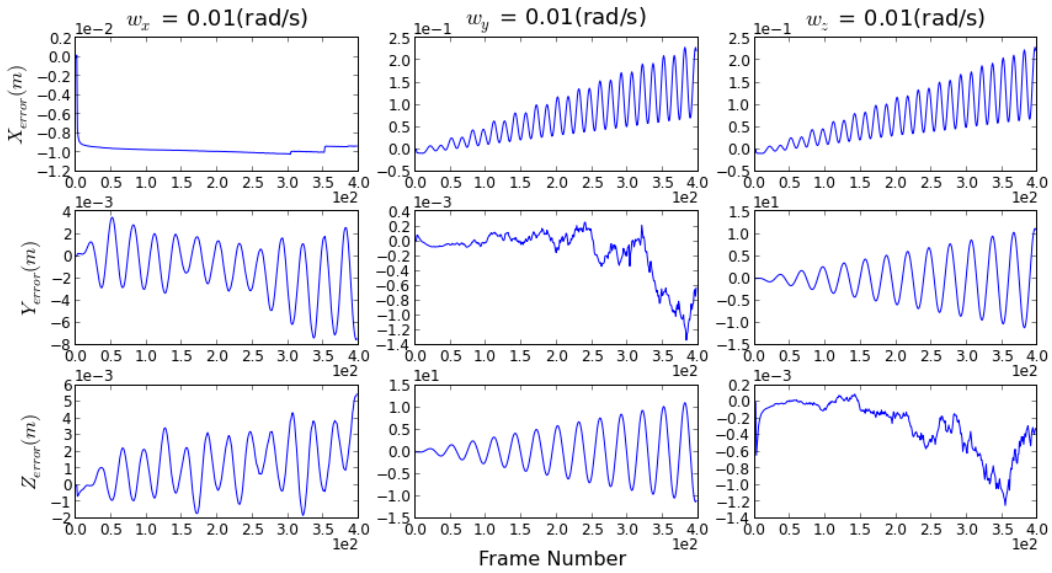

Figure 29: UAS estimated position in world frame

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 30 shows the feature mapping error statistic with added motions. 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.

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

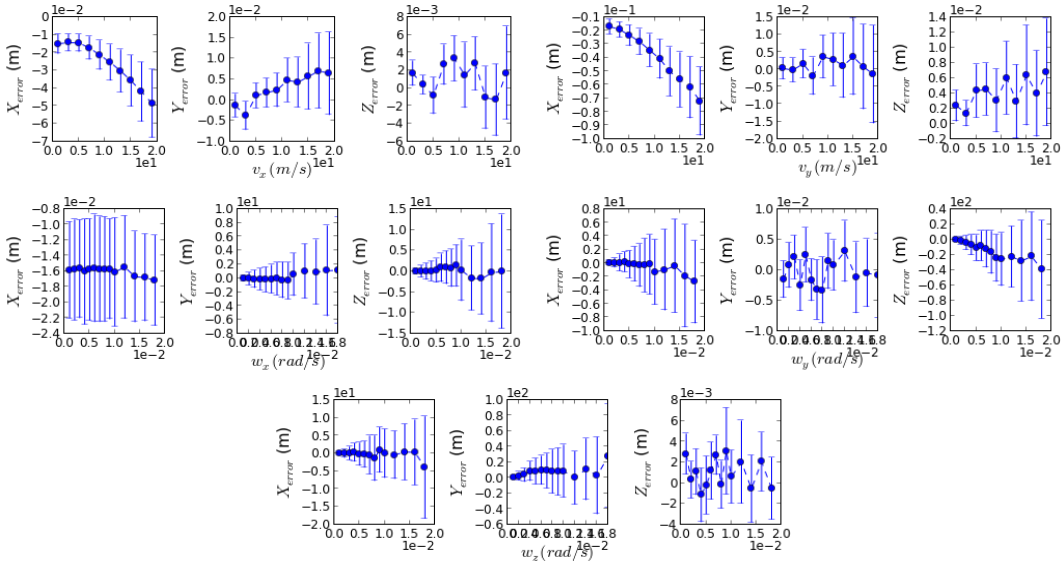

Figure 30: Feature mapping error under added motion

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 31 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.

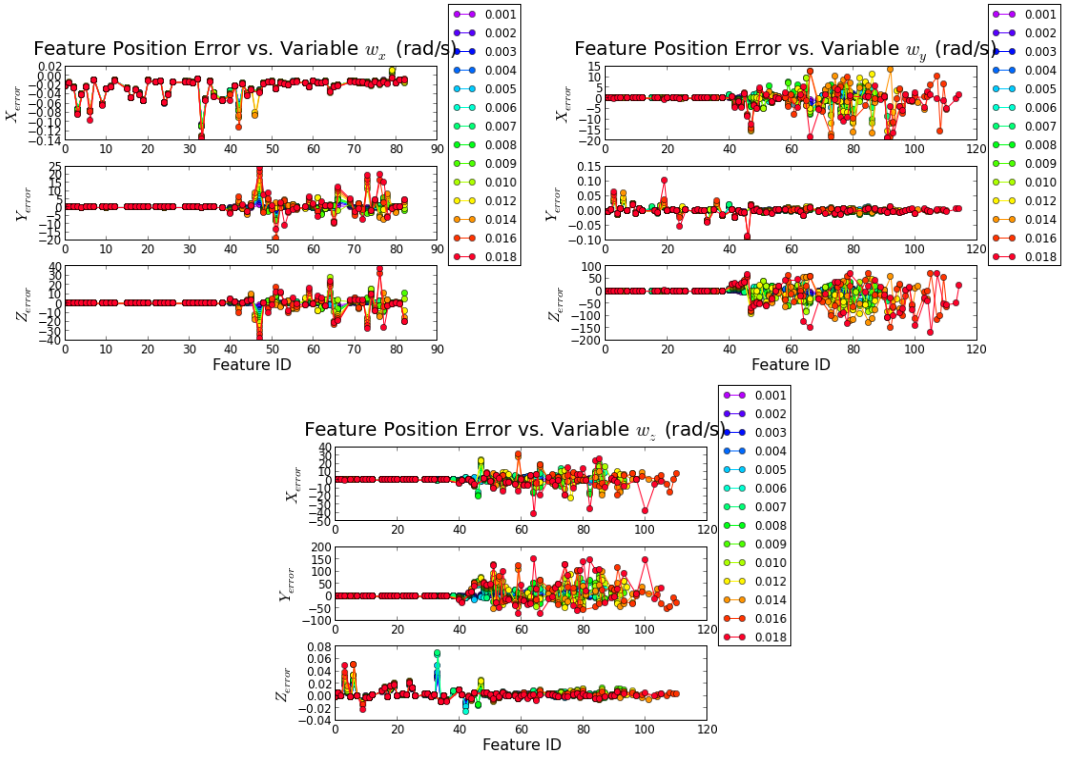

Figure 31: Feature mapping error under rotatioanl motion

- Features added after first frame has much bigger error than features added at first frame. At 1^{st} frame, 40 features 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 1 st frame with ID bigger than 40.

To investigate how does rotation motion results in bigger error on features added after first frame, feature parameters error (converted to world frame) with $wy = 0.01$ are plotted in figure 32. 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 33 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.

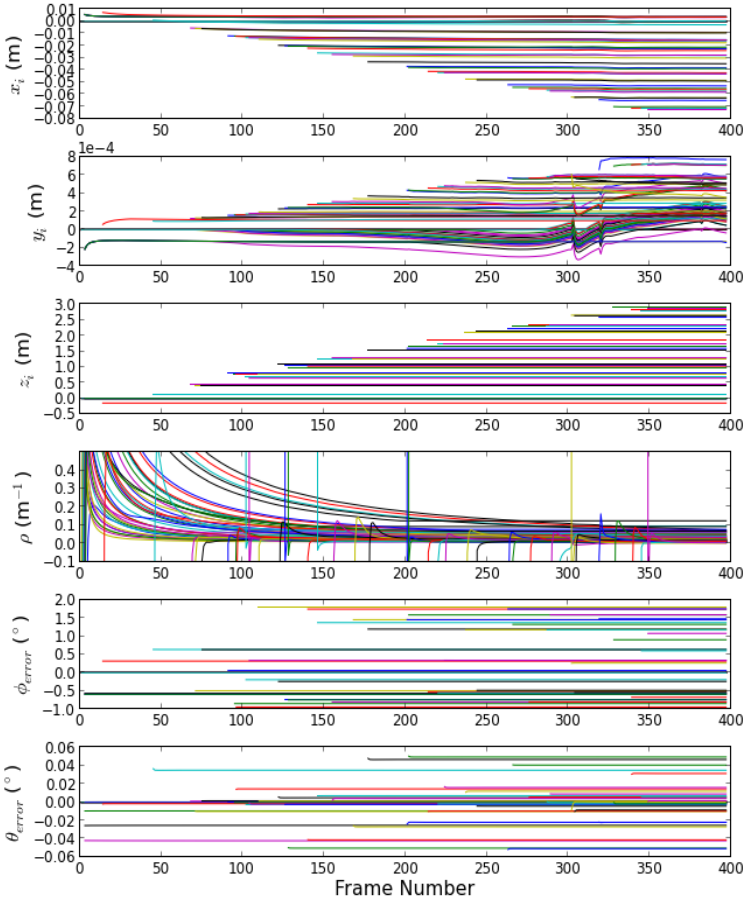

Figure 32: Feature parameters error under rotatioanl motion

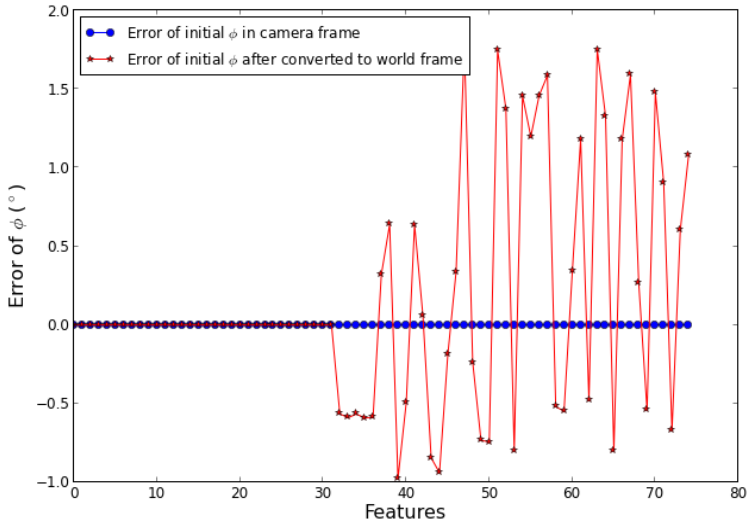

Figure 33: Error of ϕ in camera frame and world frame at initialization

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 4.2) 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 34 show an over view of UAS localization error statistics with incorrect estimates of (c_x, c_y).

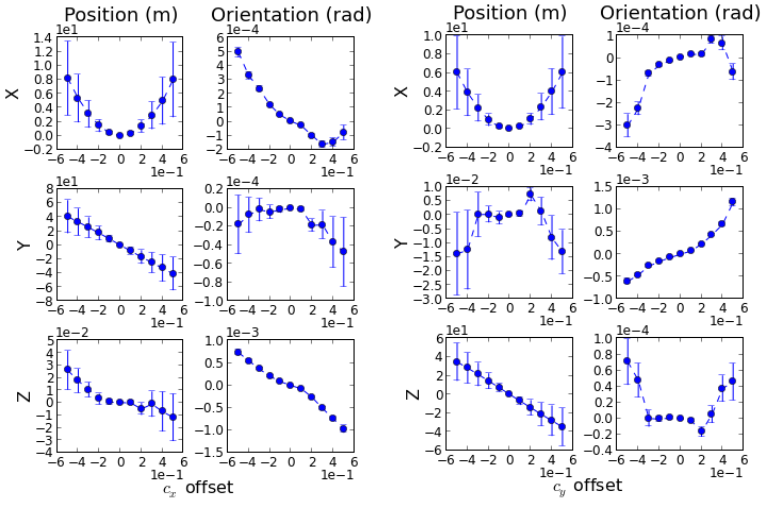

Figure 34: UAS localization error statistic with varying (c_x, c_y)

UAS position error is dependent on (c_x, c_y) and time (figure 35). The UAS position error is diverging (increases in time), and can be modeled by 1 st 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.

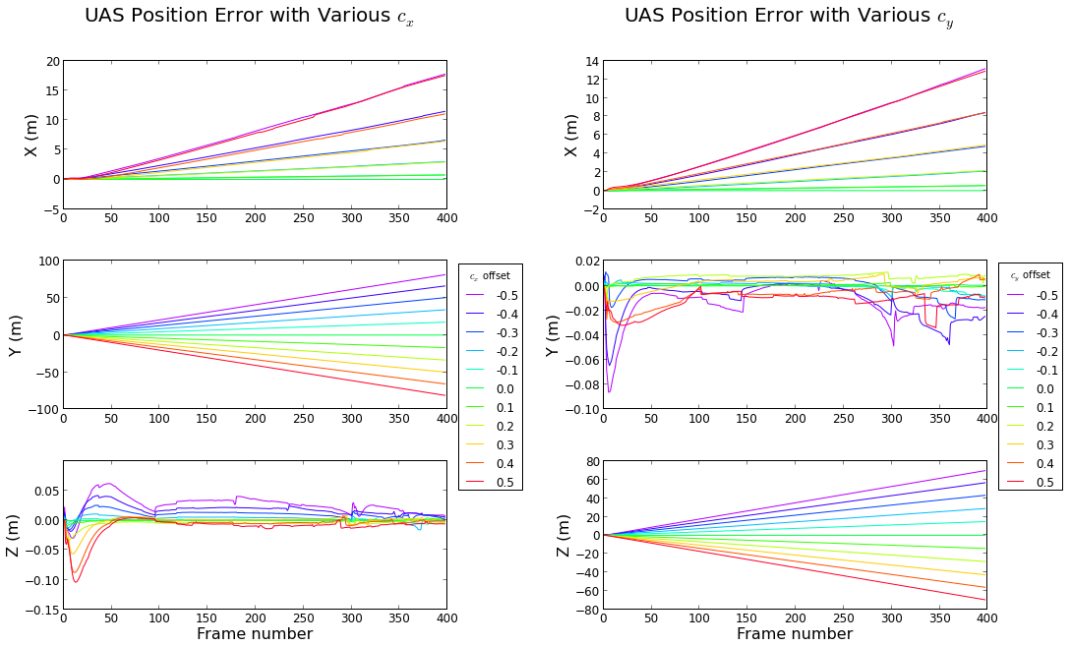

Figure 35: Diverging UAS position error

The feature mapping error statistics from incorrect estimate of (c_x, c_y) are plotted in figure 36. 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.
  - The further c_x deviate from the true value, the further the features appear (positive X axis error).
  - 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).
  - 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.

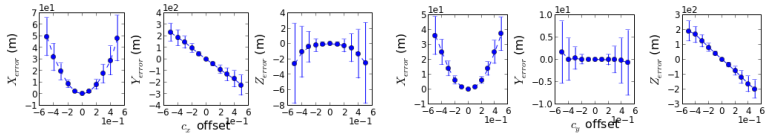

Figure 36: Feature mapping error statistic with varying (c_x, c_y)

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 37, plot (a), subplot $[1, 1]$).

- 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 37, plot (b)), 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.

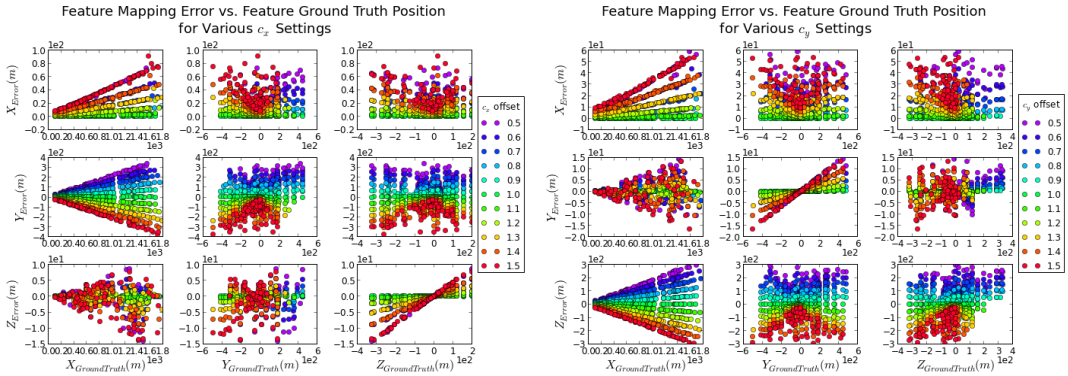

Figure 37: Feature mapping error vs. ground truth feature position

6.3.2 Effect from (f_x, f_y)

With (f_x, f_y) varying from -50% to +50% of the calibrated value, the UAS localization error is shown in 38. For all f_x and f_y settings, UAS position error remained less than +/-0.05 meters, and orientation error remained in less than 8e-6 radius. Compared to the error obtained from the ideal case simulation, Error in (f_x, f_y) estimation does not introduce any additional error into UAS localization estimate.

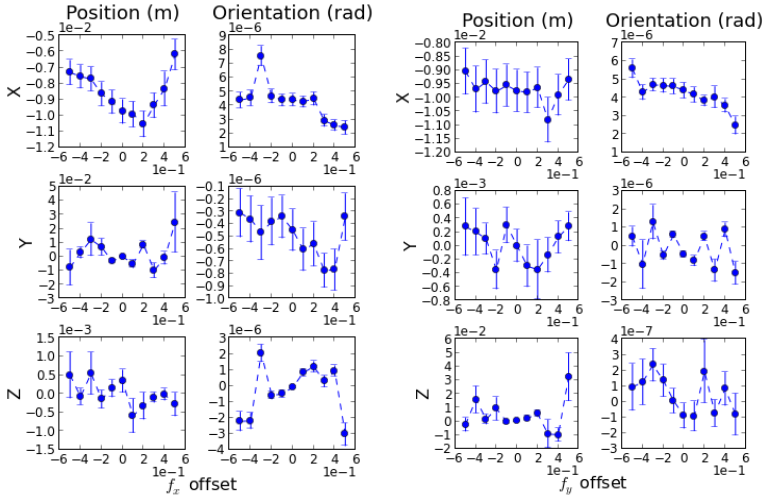

Figure 38: UAS localization error statistic with variable (f_x, f_y)

Feature mapping, on the other hand, is unavoidably affected by the error in (f_x, f_y) since these are the scaling factor that project features from 3D world onto image plane.

Figure 40 shows the error statistic of feature position estimation under variation of (f_x, f_y). The effect on the X axis component is minimum, in the scale of milli meter. Y axis component receive the most impact with unmatched f_x, since this is the scale factor that map feature's Y component in world frame onto U axis on image plane by $u = Y/X * f_x$. Same goes for the Z axis component of the feature position estimate and its relation to f_y.

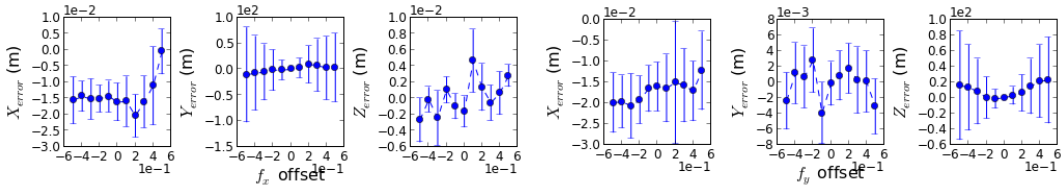

Figure 39: Feature mapping error statistic with variable (f_x, f_y)

Plotting the feature mapping error against feature ground truth position reveals how error in (f_x, f_y) estimate impact on feature position estimate. When f_x estimate contains error, Y component of feature position is determined by the feature's ground truth position on X and Y components. The value of Y_{error} is directly proportional to the Y component of its ground truth position, with the error in f_x determines the function's slope. Same relation can be found for Z_{error} with f_y, and $Z_{groundtruth}$.

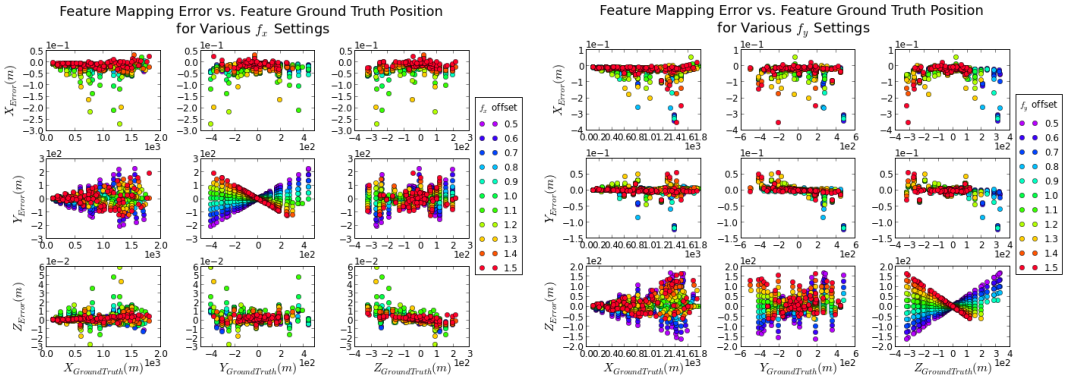

Figure 40: Feature mapping error plotted against feature ground truth for various (f_x, f_y)

6.3.3 Effect from Distortion

The CC_EKF_SLAM algorithm does not consider camera lens distortion at this stage. Therefore, the simulation evaluate the effect of distortion varying from 0% to 150% of the calibrated result to evaluate the amount error resulted by ignoring the lens distortion.

Figure 41 (left) shows the UAS localization error for all lens distortion setting. Ignoring the distortion brings significant error into the UAS localization. UAS X position receives the most impact with error up to 100 meters with increasing standard deviation. Y and Z position shows less error, but standard deviation grows larger with distortion offset.Figure 41 (right) reveals the cause of increasing standard deviation. The UAS position error was growing larger with time. The UAS orientation estimates also experience error increase going from maximum mean error of 5e-6 rad in low noise simulation to 2.5e-4 rad. The standard deviation of orientation estimates also increases with lens distortion offset. Similarly, the orientation error vs. frame number plots show that the error amplitude increase with time, although it fluctuates around zero.

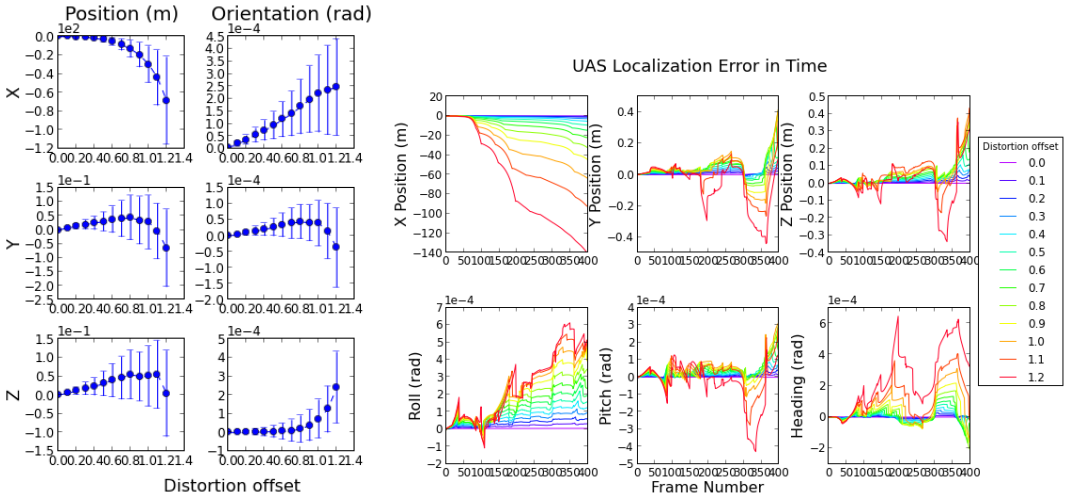

Figure 41: UAS localization error with lens distortion

The feature mapping error statistic is shown in figure 42. The X axis of feature position shows the most error, with mean value reading -400 meters with distortion offset at %150. Y, and Z axis position has less mean error, but the standard deviation is bigger. Plotting feature position error agains feature ground truth coordinate (figure 43) shows that the increasing standard deviation is due to the first order linear relation between the Y (or Z) and the Y (or Z) axis of feature ground truth coordinate, where distortion offset determine the slope of the line.

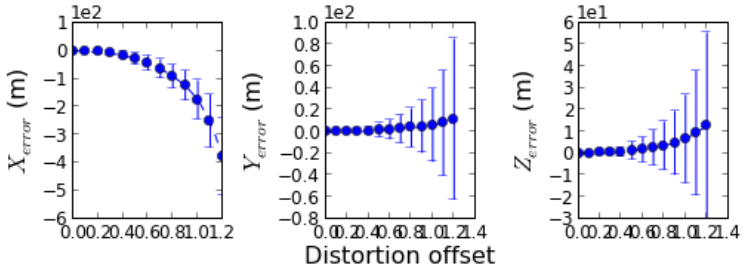

Figure 42: UAS localization error with lens distortion

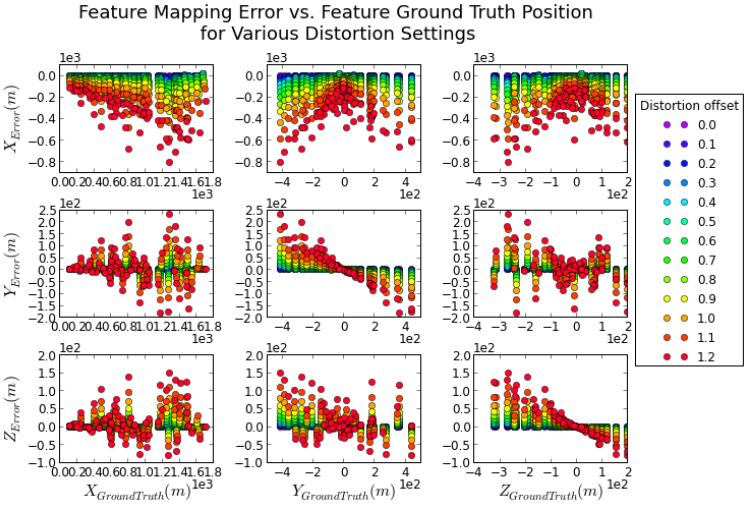

Figure 43: UAS localization error with lens distortion

6.3.4 Effect from Image Resolution

It is well known that higher resolution sensor will give more accuracy to the estimate. To know how high a resolution is good enough for the distance range that this work is targeting, a quantitive analysis is necessary. The ideal case simulation is ran with various image resolution setting, and the result is below (figure 44).

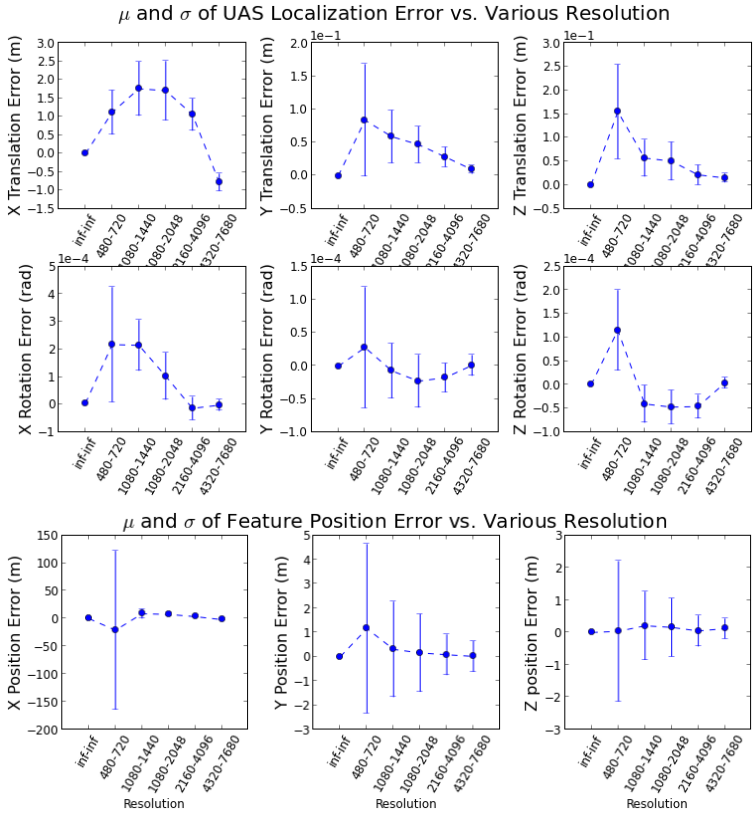

Figure 44: Error statistic for various image resolution

This test confirmed that higher resolution the image sensor is, the more accuracy it will bring. The most significant error is seen at resolution 480x720 where the X axis of feature position error is +/- 150m. At one 720x1080, which is one step up of 480x720, the X axis feature position error is hugely reduced to a few meters. For all other parameters, improvement at each level of resolution increase is nearly linear. This result suggests that to achieve reasonably good accuracy for obstacle detection,

a resolution of 720x1080 or higher is preferred.

Appendix A

Coordinate Transformation

For a mobile robot traveling in world. A point in space has coordinate $[x \quad y \quad z]$ in world frame, and $[x' \quad y' \quad z']$ in the mobile frame. The two coordinate is related by

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{World} = Q \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}_{mobile} \tag{45}$$

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 B

Jacobian Matrix for Filter Initialization

The Jacobian matrix used in state covariance matrix initialization equation (37) is derived in this section. The complete Jacobian matrix J for initializing a feature covariance is given by

$$J = \begin{bmatrix} & & & & & & 0 \\ & & & I & & & \vdots \\ & & & & & & 0 \\ \frac{\partial p_i}{\partial OX_W^C} & \frac{\partial p_i}{\partial c^C} & \frac{\partial p_i}{\partial r^C} & 0 & \dots & 0 & \frac{\partial p_i}{\partial g_i} \end{bmatrix} \tag{46}$$

Whenever a new feature is added, it's 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 OX_W^C, c^C,or r^C, therefore

$$\frac{\partial p_i}{\partial OX_W^C} = 0_{6\times 6}, \quad \frac{\partial p_i}{\partial c^C} = 0_{6\times 3}, \quad \frac{\partial p_i}{\partial r^C} = 0_{6\times 3} \tag{47}$$

J can then be simplified as

$$J = \begin{bmatrix} I & 0 \\ 0 & \frac{\partial p_i}{\partial g_i} \end{bmatrix} \tag{48}$$

whereg_i includes the variable in matrix R: $g_i = [x_i^C \quad y_i^C \quad z_i^C \quad \rho_i \quad u_i \quad v_i]$. Then

$$\frac{\partial p_i}{\partial g_i} = \begin{bmatrix} I_{3\times3} & & 0_{3\times3} & \\ & \frac{\partial \rho_i}{\partial \rho_i} & \frac{\partial \rho_i}{\partial u_i} & \frac{\partial \rho_i}{\partial v_i} \\ 0_{3\times3} & \frac{\partial \varphi_i^C}{\partial \rho_i} & \frac{\partial \varphi_i^C}{\partial u_i} & \frac{\partial \varphi_i^C}{\partial v_i} \\ & \frac{\partial \theta_i^C}{\partial \rho_i} & \frac{\partial \theta_i^C}{\partial u_i} & \frac{\partial \theta_i^C}{\partial v_i} \end{bmatrix} = \begin{bmatrix} I_{3\times3} & & 0_{3\times3} & \\ & 1 & 0 & 0 \\ 0_{3\times3} & 0 & \frac{\partial \varphi_i^C}{\partial u_i} & \frac{\partial \varphi_i^C}{\partial v_i} \\ & 0 & \frac{\partial \theta_i^C}{\partial u_i} & \frac{\partial \theta_i^C}{\partial v_i} \end{bmatrix} \tag{49}$$

Based on the rule of derivation,

$$\frac{\partial \varphi_i^C}{\partial u_i} = \frac{\partial \varphi_i^C}{\partial h^C} \frac{\partial h^C}{\partial u_i}$$

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

$$\frac{\partial \varphi_i^C}{\partial v_i} = \frac{\partial \varphi_i^C}{\partial h^C} \frac{\partial h^C}{\partial v_i}$$

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

and

$$\frac{\partial \varphi_i^C}{\partial h^C} = \begin{bmatrix} \frac{\partial \varphi_i^C}{\partial h_x^C} & \frac{\partial \varphi_i^C}{\partial h_y^C} & \frac{\partial \varphi_i^C}{\partial h_z^C} \end{bmatrix} = \begin{bmatrix} \frac{-h_x^C \cdot h_z^C}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})\sqrt{h_x^{C\,2}+h_y^{C\,2}}} \\ \frac{-h_y^W \cdot h_z^W}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})\sqrt{h_x^{C\,2}+h_y^{C\,2}}} \\ \frac{\sqrt{h_x^{C\,2}+h_y^{C\,2}}}{(h_x^{C\,2}+h_y^{C\,2}+h_z^{C\,2})} \end{bmatrix}^T \tag{50}$$

$$\frac{\partial \theta_i^C}{\partial h^C} = \begin{bmatrix} \frac{\partial \theta_i^C}{\partial h_x^C} & \frac{\partial \theta_i^C}{\partial h_y^C} & \frac{\partial \theta_i^C}{\partial h_z^C} \end{bmatrix} = \begin{bmatrix} -\frac{h_y^C}{h_x^{C\,2}+h_y^{C\,2}} & \frac{h_x^C}{h_x^{C\,2}+h_y^{C\,2}} & 0 \end{bmatrix} \tag{51}$$

$$\frac{\partial h^C}{\partial u_i} = \begin{bmatrix} \frac{\partial h_x^C}{\partial u_i} \\ \frac{\partial h_y^C}{\partial u_i} \\ \frac{\partial h_z^C}{\partial u_i} \end{bmatrix} = \begin{bmatrix} 0 \\ s_x \\ 0 \end{bmatrix} \tag{52}$$

$$\frac{\partial h^C}{\partial v_i} = \begin{bmatrix} \frac{\partial h_x^C}{\partial v_i} \\ \frac{\partial h_y^C}{\partial v_i} \\ \frac{\partial h_z^C}{\partial v_i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s_y \end{bmatrix} \tag{53}$$

Appendix C

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

$$\frac{\partial h_k^U(x)}{\partial x} = \begin{bmatrix} 0_{2n\times 6} & \frac{\partial h_k^U(x)}{\partial c^C} & \frac{\partial h_k^U(x)}{\partial r^C} & \frac{\partial h_k^U(x)}{\partial p_1} & \cdots & \frac{\partial h_k^U(x)}{\partial p_n} \end{bmatrix} = \begin{bmatrix} 0_{2n\times 6} & \frac{\partial h_k^U}{\partial h_k^C}\cdot\frac{\partial h_k^C}{\partial c^C} & \frac{\partial h_k^U}{\partial h_k^C}\cdot\frac{\partial h_k^C}{\partial r^C} & \frac{\partial h_k^U}{\partial h_k^C}\cdot\frac{\partial h_k^C}{\partial p_1} & \cdots & \frac{\partial h_k^U}{\partial h_k^C}\cdot\frac{\partial h_k^C}{\partial p_n} \end{bmatrix} \quad (54)$$

Elements in the matrix above are given in - . The formula for calculating Q^{-1}in - can be found in

$$\frac{\partial h_k^U}{\partial h_k^C} = \begin{bmatrix} -\frac{s_x h_{y,k}^C}{{h_{x,k}^C}^2} & \frac{s_x}{h_{x,k}^C} & 0 \\ -\frac{s_y h_{z,k}^C}{{h_{x,k}^C}^2} & 0 & \frac{s_y}{h_{x,k}^C} \end{bmatrix} \quad (55)$$

$$\frac{\partial h^C}{\partial c^C} = -R^{-1}(r_k^C)\rho_k \quad (56)$$

$$\frac{\partial h(x)}{\partial r^C} = \begin{bmatrix} \frac{\partial h^C}{\partial r_x^C} & \frac{\partial h^C}{\partial r_y^C} & \frac{\partial h^C}{\partial r_z^C} \end{bmatrix} \tag{57}$$

$$\frac{\partial h^C}{\partial r_x} = \begin{bmatrix} 0 \\ h_x^C(sin(r_x)sin(r_z) + cos(r_x)cos(r_z)sin(r_y)) + h_y^C(-cos(r_z)sin(r_x) + cos(r_x)sin(r_y)sin(r_z)) \\ h_x^C(cos(r_x)\sin(r_z) - cos(r_z)sin(r_x)sin(r_y)) + h_y^C(-cos(r_x)cos(r_z) - sin(r_x)sin(r_y)sin(r_z)) \end{bmatrix} \tag{58}$$

$$\frac{\partial h^C}{\partial r_y} = \begin{bmatrix} -h_z^C\cos(r_y) - h_x^C\cos(r_z)sin(r_y) - h_y^C sin(r_y)sin(r_z) \\ -h_z^C sin(r_x)sin(r_y) + h_x^C cos(r_y)cos(r_z)sin(r_x) + h_y^C cos(r_y)sin(r_x)sin(r_z) \\ -h_z^C cos(r_x)sin(r_y) + h_x^C cos(r_x)cos(r_y)cos(r_z) + h_y^C cos(r_x)cos(r_y)sin(r_z) \end{bmatrix} \tag{59}$$

$$\frac{\partial h^C}{\partial r_z} = \begin{bmatrix} h_y^C cos(r_y)cos(r_z) - h_x^C cos(r_y)sin(r_z) \\ h_x^C(-cos(r_x)\cos(r_z) - sin(r_x)\sin(r_y)\sin(r_z)) + h_y^C(-cos(r_x)sin(r_z) + cos(r_z)sin(r_x)sin(r_y \\ h_x^C(cos(r_z)sin(r_x) - cos(r_x)sin(r_y)sin(r_z)) + h_y^C(sin(r_x)sin(r_z) + cos(r_x)cos(r_z)sin(r_y)) \end{bmatrix} \tag{60}$$

$$\frac{\partial h^C}{\partial p_i} = \begin{bmatrix} \frac{\partial h^C}{\partial x_i^C} & \frac{\partial h^C}{\partial y_i^C} & \frac{\partial h^C}{\partial z_i^C} & \frac{\partial h^C}{\partial \rho_i} & \frac{\partial h^C}{\partial \theta_i} & \frac{\partial h^C}{\partial \varphi_i} \end{bmatrix} \tag{61}$$

Equation above is a 3x6 matrix

$$\begin{bmatrix} \frac{\partial h^C}{\partial x_i^C} & \frac{\partial h^C}{\partial y_i^C} & \frac{\partial h^C}{\partial z_i^C} \end{bmatrix} = Q^{-1}(r_k^C)\rho_k \tag{62}$$

$$\frac{\partial h^C}{\partial \rho_i} = -Q^{-1}(r_k^C) \cdot c_k \tag{63}$$

$$\frac{\partial h^C}{\partial \theta_i} = Q^{-1}(r_k^C) \begin{bmatrix} -cos(\varphi_i)\sin(\theta_i) \\ \cos(\varphi_i)cos(\theta_i) \\ 0 \end{bmatrix} \tag{64}$$

$$\frac{\partial h^C}{\partial \varphi_i} = Q^{-1}(r_k^C) \begin{bmatrix} -cos(\varphi_i)sin(\theta_i) \\ -sin(\varphi_i)sin(\theta_i) \\ cos(\varphi_i) \end{bmatrix} \tag{65}$$

Jacobian Matrix of Composition Equations

$$J_{C_{k-1}\to C_k} = \frac{\partial x_k}{\partial x_{k-1}} = \begin{bmatrix} \frac{\partial x_k}{\partial OX_{W,k-1}^C} & \frac{\partial x_k}{\partial c_{k-1}} & \frac{\partial x_k}{\partial r_{k-1}} & \frac{\partial x_k}{\partial p_{1,k-1}^C} & \frac{\partial x_k}{\partial p_{2,k-1}^C} & \cdots \end{bmatrix} \tag{66}$$

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

$$\frac{\partial x_k}{\partial OX_{W,k-1}^C} = \begin{bmatrix} \frac{\partial O_{W,k}^C}{\partial O_{W,k-1}^C} & \frac{\partial O_{W,k}^C}{\partial W_{W,k-1}^C} \\ \frac{\partial W_{W,k}^C}{\partial O_{W,k-1}^C} & \frac{\partial W_{W,k}^C}{\partial W_{W,k-1}^C} \\ \frac{\partial c_k}{\partial O_{W,k-1}^C} & \frac{\partial c_k}{\partial W_{W,k-1}^C} \\ \frac{\partial r_k}{\partial O_{W,k-1}^C} & \frac{\partial r_k}{\partial W_{W,k-1}^C} \\ \frac{\partial p_i^C}{\partial O_{W,k-1}^C} & \frac{\partial p_i^C}{\partial W_{W,k-1}^C} \end{bmatrix} = \begin{bmatrix} Q^{-1}(r^C) & 0 \\ 0 & I_{3\times 3} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{67}$$

Derivatives of $x_k$with respect to c and $p$are given in -

$$\begin{bmatrix} \frac{\partial O^C_{W,k}}{\partial c_{k-1}} & \frac{\partial O^C_{W,k}}{\partial r_{k-1}} \\ \frac{\partial W^C_{W,k}}{\partial c_{k-1}} & \frac{\partial W^C_{W,k}}{\partial r_{k-1}} \\ \frac{\partial c_k}{\partial c_{k-1}} & \frac{\partial c_k}{\partial r_{k-1}} \\ \frac{\partial r_k}{\partial c_{k-1}} & \frac{\partial r_k}{\partial r_{k-1}} \\ \frac{\partial p(0:2)^C_i}{\partial c_{k-1}} & \frac{\partial p(0:2)^C_i}{\partial r_{k-1}} \\ \frac{\partial p(3:5)^C_i}{\partial c_{k-1}} & \frac{\partial p(3:5)^C_i}{\partial r_{k-1}} \end{bmatrix} = \begin{bmatrix} -Q^{-1}(r_k^{C_{k-1}}) & \frac{\partial Q^{-1}(r_k^{C_{k-1}})}{\partial r_{k-1}} \\ 0_{3\times3} & -I_{3\times3} \\ I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \\ -Q^{-1}(r_k^{C_{k-1}}) & \frac{\partial Q^{-1}(r_k^{C_{k-1}})}{\partial r_{k-1}} \\ 0_{3\times3} & pr \end{bmatrix} \tag{68}$$

where

$$pr = \begin{bmatrix} 1 & 0 & 0 \\ & \frac{\partial \varphi^C_{i,k}}{\partial r_{k-1}} & \\ & \frac{\partial \theta^C_{i,k}}{\partial r_{k-1}} & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & \frac{\partial m^{-1}(Q^{-1}(r_k^{C_{k-1}})m(\varphi^{C_{k-1}}_{i,k},\theta^{C_{k-1}}_{i,k}))}{\partial r_{k-1}} & \end{bmatrix} \tag{69}$$

Let $M = Q^{-1}(r_k^{C_{k-1}})m(\varphi^{C_{k-1}}_{i,k}, \theta^{C_{k-1}}_{i,k})$, and $m^{-1} = m^{-1}(M)$, then

$$\frac{\partial m^{-1}}{\partial r_k^{C_{k-1}}} = \frac{\partial m^{-1}}{\partial M} \cdot \frac{\partial M}{\partial r_k^{C_{k-1}}} \tag{70}$$

where

$$\frac{\partial M}{\partial r_k^{C_{k-1}}} = \frac{\partial Q^{-1}(r_k^{C_{k-1}})}{\partial r_{k-1}} \cdot m(\varphi^{C_{k-1}}_{i,k}, \theta^{C_{k-1}}_{i,k}) \tag{71}$$

as m^{-1} isthe function that calculate $\begin{bmatrix} \varphi & \theta \end{bmatrix}$ from a vector,$\frac{\partial m^{-1}}{\partial M}$ is the same as and .

Derivatives of x_k with respect to feature parameters $p_i^C$are given in

$$\frac{\partial x_k^{C_k}}{\partial p_{i,k}^{C_{k-1}}} = \begin{bmatrix} \frac{\partial O_{W,k}^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial W_{W,k}^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial c_k^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial r_k^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial p_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{\partial p_{i,k}^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \end{bmatrix} \tag{72}$$

$$\frac{\partial p_{i,k}^{C_k}}{\partial p_{i,k}^{C_{k-1}}} = \begin{bmatrix} \frac{\partial x_{i,k}^{C_k}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial y_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial z_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial \rho_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial \varphi_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \\ \frac{\partial \theta_{i,k}^{C}}{\partial p_{i,k}^{C_{k-1}}} \end{bmatrix} = \begin{bmatrix} Q^{-1}(r_k^{C_{k-1}}) & 0_{3\times 3} \\ 0_{3\times 3} & \frac{\partial p(3:5)_{i,k}^{C_k}}{\partial p(3:5)_{i,k}^{C_{k-1}}} \end{bmatrix} \tag{73}$$

$$\frac{\partial p(3:5)_{i,k}^{C_k}}{\partial p(3:5)_{i,k}^{C_{k-1}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\partial \varphi_{i,k}^{C_k}}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \varphi_{i,k}^{C_k}}{\partial \theta_{i,k}^{C_{k-1}}} \\ 0 & \frac{\partial \theta_{i,k}^{C_k}}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \theta_{i,k}^{C_k}}{\partial \theta_{i,k}^{C_{k-1}}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\partial \varphi_{i,k}^{C_k}}{\partial M}\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \varphi_{i,k}^{C_k}}{\partial M}\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}} \\ 0 & \frac{\partial \theta_{i,k}^{C_k}}{\partial M}\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}} & \frac{\partial \theta_{i,k}^{C_k}}{\partial M}\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}} \end{bmatrix} \tag{74}$$

in which $\frac{\partial \varphi_{i,k}^{C_k}}{\partial M}$ is the same as equation **??**. $\frac{\partial \theta_{i,k}^{C_k}}{\partial M}$ is the same as equation **??** . $\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}}$ and $\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}}$ is given below.

$$\frac{\partial M}{\partial \varphi_{i,k}^{C_{k-1}}} = R^{-1}(r^C) \begin{bmatrix} -\cos(\theta_{i,k}^{C_{k-1}}) \sin(\varphi_{i,k}^{C_{k-1}}) \\ -\sin(\varphi_{i,k}^{C_{k-1}}) \sin(\theta_{i,k}^{C_{k-1}}) \\ \cos(\varphi_{i,k}^{C_{k-1}}) \end{bmatrix} \tag{75}$$

$$\frac{\partial M}{\partial \theta_{i,k}^{C_{k-1}}} = R^{-1}(r^C) \begin{bmatrix} -\cos(\varphi_{i,k}^{C_{k-1}}) \sin(\theta_{i,k}^{C_{k-1}}) \\ \cos(\varphi_{i,k}^{C_{k-1}}) \cos(\theta_{i,k}^{C_{k-1}}) \\ 0 \end{bmatrix} \tag{76}$$

List of References

[1] T. James. "GeoSurv II UAV system requirements document." Technical report (2008).

[2] A. De Angelis, M. Dionigi, A. Moschitta, and P. Carbone. "A low-cost ultra-wideband indoor ranging technique." In "IEEE Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007," pages 1–6 (2007).

[3] 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* **58**(2), 318–329. ISSN 0018-9456 (2009).

[4] M. Harb, R. Abielmona, K. Naji, and E. Petriu. "Neural networks for environmental recognition and navigation of a mobile robot." In "IEEE Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008," pages 1123–1128 (2008).

[5] M. M. Saad, C. J. Bleakley, and S. Dobson. "Robust high-accuracy ultrasonic range measurement system." *IEEE Transactions on Instrumentation and Measurement* **60**(10), 3334–3341. ISSN 0018-9456 (2011).

[6] S. B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems* (2001).

[7] K. S. Chong and L. Kleeman. *Feature-based Mapping in Real, Large Scale Environments using an Ultrasonic Array* (1999).

[8] E. Hanna, P. Straznicky, and R. Goubran. "Obstacle detection for low flying unmanned aerial vehicles using stereoscopic imaging." In "IEEE Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008," pages 113–118 (2008).

[9] 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 "2010 IEEE Instrumentation and Measurement Technology Conference (I2MTC)," pages 318–322 (2010).

[10] J. Civera, A. Davison, and J. Montiel. "Inverse depth parametrization for monocular SLAM." *IEEE Transactions on Robotics* **24**(5), 932–945. ISSN 1552-3098 (2008).

[11] R. Jirawimut, S. Prakoonwit, F. Cecelja, and W. Balachandran. "Visual odometer for pedestrian navigation." *IEEE Transactions on Instrumentation and Measurement* **52**(4), 1166–1173. ISSN 0018-9456 (2003).

[12] F. Amzajerdian, D. Pierrottet, L. Petway, G. Hines, and V. Roback. "Lidar systems for precision navigation and safe landing on planetary bodies." volume 8192, pages 819202–819202–7 (2011).

[13] R. Subharsanan and R. Moss. "Low cost scanning LiDAR imager." (2013).

[14] M. Lemmens. "Airborne LiDAR sensors." (2007).

[15] E. Einhorn, C. Schrter, and H. M. Gross. "Can't take my eye off you: Attention-driven monocular obstacle detection and 3D mapping." In "2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)," pages 816–821 (2010).

[16] H. Hashimoto, T. Yamaura, and M. Higashiguchi. "Detection of obstacle from monocular vision based on histogram matching method." In ", Proceedings of the 1996 IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation, 1996," volume 2, pages 1047–1051 vol.2 (1996).

[17] K. Yamaguchi, T. Kato, and Y. Ninomiya. "Moving obstacle detection using monocular vision." In "2006 IEEE Intelligent Vehicles Symposium," pages 288–293 (2006).

[18] F. Zhang, R. Goubran, and P. Straznicky. "Obstacle detection for low flying UAS using monocular camera." In "Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International," pages 2133–2137 (2012).

[19] D. Maier, M. Bennewitz, and C. Stachniss. "Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data." In "2011 IEEE International Conference on Robotics and Automation (ICRA)," pages 1263–1269 (2011).

[20] S. Kubota, T. Nakano, and Y. Okamoto. "A global optimization algorithm for real-time on-board stereo obstacle detection systems." In "2007 IEEE Intelligent Vehicles Symposium," pages 7–12 (2007).

[21] 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 "2009 IEEE Intelligent Vehicles Symposium," pages 36–41 (2009).

[22] Z. Zhang, Y. Wang, J. Brand, and N. Dahnoun. "Real-time obstacle detection based on stereo vision for automotive applications." In "Education and Research Conference (EDERC), 2012 5th European DSP," pages 281–285 (2012).

[23] W. Van der Mark, J. Van Den Heuvel, and F. C. A. Groen. "Stereo based obstacle detection with uncertainty in rough terrain." In "2007 IEEE Intelligent Vehicles Symposium," pages 1005–1012 (2007).

[24] A. Broggi, M. Buzzoni, M. Felisa, and P. Zani. "Stereo obstacle detection in challenging environments: The VIAC experience." In "2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)," pages 1599–1604 (2011).

[25] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc. ISBN 9780596554040 (2008).

[26] J. Heikkila and O. Silven. "A four-step camera calibration procedure with implicit image correction." In ", 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings," pages 1106–1112 (1997).

[27] R. Tsai. "An efficient and accurate camera calibration technique for 3D machine vision." In "IEEE Conference on Computer Vision and Pattern Recognition," pages 364–374. Miami Beach, FL. (1986).

[28] Z. Zhang. "A flexible new technique for camera calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11), 1330–1334. ISSN 0162-8828 (2000).

[29] R. E. Kalman. "A new approach to linear filtering and prediction problems." *Journal of Basic Engineering* **82**(1), 35–45. ISSN 0098-2202 (1960).

[30] H. Sorenson. "Least-squares estimation: from gauss to kalman." *IEEE Spectrum* **7**(7), 63–68. ISSN 0018-9235 (1970).

[31] Analytic Sciences Corporation. *Applied optimal estimation*. M.I.T. Press, Cambridge, Mass. ISBN 0262200279 (1974).

[32] M. S. Grewal. *Kalman filtering: theory and practice*. Prentice-Hall information and system sciences series. Prentice-Hall, Englewood Cliffs, N.J. ISBN 013211335X (1993).

[33] F. L. Lewis. *Optimal estimation: with an introduction to stochastic control theory*. Wiley, New York. ISBN 0471837415 (1986).

[34] R. G. Brown. *Introduction to random signals and applied Kalman filtering*. John Wiley & Sons, Inc., New York, 2nd ed edition. ISBN 0471525731 (1993).

[35] P. S. Maybeck. *Stochastic models, estimation and control*. Number v.141 in Mathematics in science and engineering. Academic Press, New York. ISBN 0124807011 (1979).

[36] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part i." *IEEE Robotics Automation Magazine* **13**(2), 99–110. ISSN 1070-9932 (2006).

[37] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II." *IEEE Robotics Automation Magazine* **13**(3), 108–117. ISSN 1070-9932 (2006).

[38] R. C. Smith and P. Cheeseman. "On the representation and estimation of spatial uncertainty." *The International Journal of Robotics Research* **5**(4), 56–68. ISSN 0278-3649, 1741-3176 (1986).

[39] H. Durrant-Whyte. "Uncertain geometry in robotics." *IEEE Journal of Robotics and Automation* **4**(1), 23–31. ISSN 0882-4967 (1988).

[40] J. Leonard and H. Durrant-Whyte. "Simultaneous map building and localization for an autonomous mobile robot." In "IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91," pages 1442–1447 vol.3 (1991).

[41] H. Durrant-Whyte, D. Rye, and E. Nebot. "Localization of autonomous guided vehicles." In G. Giralt and G. H. P. Dr.-Ing, editors, "Robotics Research," pages 613–625. Springer London. ISBN 978-1-4471-1257-0, 978-1-4471-1021-7 (1996).

[42] M. Csorba, J. K. Uhlmann, and H. F. Durrant-Whyte. "New approach to simultaneous localization and dynamic map building." In "Proc. SPIE 2738, Navigation and Control Technologies for Unmanned Systems," volume 2738, pages 26–36. Orlando, FL (1996).

[43] M. Csorba. *Simultaneous Localisation and Map Building* (1997).

[44] 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* **17**(3), 229–241. ISSN 1042-296X (2001).

[45] A. Martinelli, N. Tomatis, and R. Siegwart. "Some results on SLAM and the closing the loop problem." In "2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)," pages 2917–2922 (2005).

[46] S. Julier and J. Uhlmann. "A counter example to the theory of simultaneous localization and map building." In "IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA," volume 4, pages 4238–4243 vol.4 (2001).

[47] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. "Consistency of the EKF-SLAM algorithm." In "2006 IEEE/RSJ International Conference on Intelligent Robots and Systems," pages 3562–3568 (2006).

[48] U. Frese. "A discussion of simultaneous localization and mapping." *Autonomous Robots* **20**(1), 25–42. ISSN 0929-5593, 1573-7527 (2006).

[49] J. A. Castellanos, J. Neira, and J. D. Tards. "Limits to the consistency of ekf-based slam." In "5th IFAC Symp. Intell. Autonom. Veh., IAV04," Lisbon, Portugal (2004).

[50] G. P. Huang, A. Mourikis, and S. Roumeliotis. "Analysis and improvement of the consistency of extended kalman filter based SLAM." In "IEEE International Conference on Robotics and Automation, 2008. ICRA 2008," pages 473–479 (2008).

[51] S. Huang and G. Dissanayake. "Convergence and consistency analysis for extended kalman filter based SLAM." *IEEE Transactions on Robotics* **23**(5), 1036–1049. ISSN 1552-3098 (2007).

[52] 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 2009 IEEE/RSJ international conference on Intelligent robots and systems," IROS'09, page 34983504. IEEE Press, Piscataway, NJ, USA. ISBN 978-1-4244-3803-7 (2009).

[53] C. Estrada, J. Neira, and J. Tardos. "Hierarchical SLAM: real-time accurate mapping of large environments." *IEEE Transactions on Robotics* **21**(4), 588–596. ISSN 1552-3098 (2005).

[54] J. Leonard and P. Newman. "Consistent, convergent, and constant-time slam." In "In IJCAI," (2003).

[55] M. Bosse, P. Newman, J. Leonard, and S. Teller. "SLAM in large-scale cyclic environments using the atlas framework." *International Journal of Robotics Research* page 11131139 (2004).

[56] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments* (2002).

[57] "Python."

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

[59] J. Y. Bouguet *et al.* "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm." *Intel Corporation, Microprocessor Research Labs, OpenCV Documents* (1999).

[60] "Athena 111m integrated flight control system."

[61] "CGIAR-CSI SRTM 90m DEM digital elevation database."

[62] "World geodetic system - wikipedia, the free encyclopedia."

[63] "pyproj - python interface to PROJ.4 library - google project hosting."

[64] "GPS accuracy."