TRACKING GROUND TARGETS WITH MEASUREMENTS OBTAINED FROM A SINGLE MONOCULAR CAMERA MOUNTED ON AN UNMANNED AERIAL VEHICLE

by

DUSTIN DENEAULT

B.S., Kansas State University, 2006

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical and Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2007

Approved by:

Major Professor
Dr. Dale Schinstock

Abstract

The core objective of this research is to develop an estimator capable of tracking the states of ground targets with observation measurements obtained from a single monocular camera mounted on a small unmanned aerial vehicle (UAV). Typical sensors on a small UAV include an inertial measurement unit (IMU) with three axes accelerometer and rate gyro sensors and a global positioning system (GPS) receiver which gives position and velocity estimates of the UAV. Camera images are combined with these measurements in state estimate filters to track ground features of opportunity and a target. The images are processed by a keypoint detection and matching algorithm that returns pixel coordinates for the features. Kinematic state equations are derived that reflect the relationships between the available input and output measurements and the states of the UAV, features, and target. These equations are used in the development of coupled state estimators for the dynamic state of the UAV, for estimation of feature positions, and for estimation of target position and velocity.

The estimator developed is tested in MATLAB/SIMULINK, where GPS and IMU data are generated from the simulated states of a nonlinear model of a Navion aircraft. Images are also simulated based upon a fabricated environment consisting of features and a moving ground target. Target observability limitations are overcome by constraining the target vehicle to follow ground terrain, defined by local features, and subsequent modification of the target's observation model. An unscented Kalman filter (UKF) provides the simultaneous localization and mapping solution for the estimation of aircraft states and feature locations. Another filter, a loosely coupled Kalman filter for the target states, receives 3D measurements of target position with estimated covariance obtained by an unscented transformation (UT). The UT uses the mean and covariance from the camera measurements and from the UKF estimated aircraft states and feature locations to determine the estimated target mean and covariance. Simulation results confirm that the new loosely coupled filters are capable of estimating target states. Experimental data, collected from a research UAV, explores the effectiveness of the terrain estimation techniques required for target tracking.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to personally thank my Major Professor, Dr. Dale Schinstock, for providing me with the opportunity to work in the AVS Lab at KSU. I have found the overall experience to be challenging, exciting, and very rewarding. I would also like to acknowledge Dr. Chris Lewis for his continual help with AVS Lab affairs, including personal research assistance and conference paper collaboration. Also, many thanks go to Dr. Warren White for the mentoring, both as a teacher and employer, he has provided me from the time I started school at KSU.

Finally, I need to thank Craig Buckley for his dedicated and persistent programming of the PC104 Stack and integration of the digital camera. His work helped collect the experimental data shown in this paper.

Dedication

I dedicate this paper to all of those who have ever given me a chance, including teachers, coaches, friends, and family members.

I also want to personally dedicate this paper to Jennifer, who has and will continue to make me a better person.

Acronyms

ARF	Almost Ready to Fly
AVS	Autonomous Vehicle Systems
BEC	Battery Elimination Circuit
CAN	Controller Area Network
CCT	Cloud Cap Technology, Inc.
DGPS	Differentially Corrected Global Positioning System
DOF	Degree of Freedom
EKF	Extended Kalman Filter
FOV	Field of View
GPIO	General Purpose Input Output
GPS	Global Positioning System
GPS/INS	Global Positioning System aided Inertial Navigation Solution
IDE	Integrated Drive Electronics
IMU	Inertial Measurement Unit
INS	Inertial Navigation Solution
KF	Kalman Filter
KSU	Kansas State University
LC	Loosely Coupled
LIDAR	Light Detection and Ranging
NED	North East Down
OI	Operator Interface
PDF	Probability Density Function
PWM	Pulse Width Modulation
RAM	Random Access Memory
RC	Remote Control
RV	Random Variables
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
USB	Universal Serial Bus
UT	Unscented Transformation
2D	Two Dimensional
3D	Three Dimensional

Nomenclature

Filter Variables

$\hat{\vec{x}}$	State Estimate
$\hat{\vec{x}}^a$	Augmented State Estimate
$\hat{\vec{z}}$	Output Estimate
$\boldsymbol{\chi}$	Sigma Points
$\boldsymbol{\chi}^a$	Augmented Sigma Points
$\boldsymbol{\zeta}$	Output Sigma Points
$\vec{W}$	Weight of Sigma Points
$\tilde{\vec{u}}$	Corrupted Input Measurement
$\tilde{\vec{z}}$	Corrupted Output Measurement
$\vec{w}$	White Input Noise
$\vec{v}$	White Output Noise
N_x	Size of State Vector
$\mathbf{P}^{xx}$	State Covariance
$\mathbf{P}^{a}$	Augmented State Covariance
$\mathbf{P}^{zz}$	Innovation Covariance
$\mathbf{P}^{xz}$	Cross Covariance
$\mathbf{Q}$	Input Noise Covariance
$\mathbf{R}$	Output Noise Covariance
$\mathbf{K}$	Kalman Gain
$\mathbf{\Phi}$	State Transition Model
$\mathbf{\Gamma}$	Control Input Model

Model Variables

$\vec{p}_{a\{i\}}$	Inertial Frame Aircraft Position
$\dot{\vec{p}}_{a\{i\}}$	Inertial Frame Aircraft Velocity
$\vec{q}$	Quaternion
$\vec{p}_{f\{i\}}$	Inertial Frame Feature Position
$\vec{i}_{f\{b\}}$	Body Frame Feature Image Space Pixel Coordinates
$\vec{p}_{t\{i\}}$	Inertial Frame Target Position
$\dot{\vec{p}}_{t\{i\}}$	Inertial Frame Target Velocity
$\vec{i}_{t\{b\}}$	Body Frame Target Image Space Pixel Coordinates
$\mathbf{R}_{\{ib\}}$	Body Frame to Inertial Frame Rotation Matrix
$\mathbf{R}_{\{bi\}}$	Inertial Frame to Body Frame Rotation Matrix

Subscripts and Superscripts

$(\cdot)_{\{b\}}$ - Body Frame Coordinate System

$(\cdot)_{\{i\}}$ - Inertial Frame Coordinate System

$(\cdot)_{m}$ - Measurement

$(\bar{\cdot})$ - Mean

$(\hat{\cdot})$ - State Estimate

$(\tilde{\cdot})$ - White Noise Corrupted

$(\cdot)_{k-1|k-1}$ - Corrected or Final Value from Previous Time Step

$(\cdot)_{k|k-1}$ - Predicted Value for Current Time Step

$(\cdot)_{k|k}$ - Corrected or Final Value for Current Time Step

Chapter 1: Introduction

This thesis details the development of an estimation technique which tracks the position and velocity of ground targets with measurements obtained from a single monocular camera mounted on an unmanned aerial vehicle (UAV). The tracking is limited to ground targets as this constraint is necessary to deal with observability limitations associated with single vision target tracking approaches. The well known simultaneous localization and mapping (SLAM) estimation technique is also applied in estimating the states of the UAV and local ground features defining the environment since it is required as an intermediate step for the solution of this complex problem. SLAM refers to the class of optimal state estimation methods in which both vehicle states and the environment about the vehicle are simultaneously estimated within a probabilistic framework. Specifically, knowledge of the relationships between the UAV and the ground features, observed as distinctive gradient changes in images, aid in the development of a solution for the aircraft states and a digital terrain map of the UAV's environment.

1.1 Previous Work

Many disciplines have investigated solutions to the target tracking problem for various applications including tracking military convoys, air traffic monitoring, and surveillance systems. Much of this work utilizes range and bearing sensors such as radar or laser scanners. For example, both [1] and [7] modified aircraft tracking algorithms to incorporate radar measurements taken from stationary positions. While these algorithms are readily extended to tracking ground vehicles, the targets must be located in the vicinity of the measurement device. For many situations, this may be simply impractical depending upon the nature of the targets (e.g. friend or enemy). Ref. [14] provides an alternative to fixed sensor locations by assuming moving target indicator reports, or target position measurements, are available from an aircraft sensor. Even though the author limits this approach by only considering ground vehicles moving in a 2D plane, the use of aircraft in tracking ground targets provides several advantages over fixed radar stations. Many aerial vehicles have a tremendous range of operation, are not limited by ground topography, and are often flown autonomously. In recent years, the use of UAVs for missions which require great precision or are too dangerous for manned flight has increased due to their autonomous capabilities. However, many smaller framed UAVs are generally not equipped with either radar or laser range finders due to weight and power limitations. For such power and weight limited systems, cameras are the sensor of choice.

UAV mounted vision systems are appealing for several reasons and have been applied in a variety of applications including geophysical surveying, remote sensing, ecological research, and autonomous navigation. In general, cameras are lightweight, inexpensive, and provide informative data that can be utilized in many different ways. While it is possible to obtain range measurement from stereoscopic cameras, this is not a viable option on UAVs because the stereoscopic effects are limited by the large distance the camera is located from the terrain. Therefore, monocular cameras are predominately used aboard aircraft. The use of cameras in tracking targets is also an emerging UAV research field. In target tracking applications, cameras have an advantage over several ranging sensors, which are typically active and allow the target to know when it is being observed.

However, several implications arise from the inherit projection of 3D space onto 2D image space. The inability to resolve scale for scene reconstruction from images alone is well known within machine vision literature. Recovery of scale requires other sensors such as the global positioning system (GPS) that provide absolute position or control points with known locations within the scene. Even with GPS sensors, this loss of scale continues to pose observability issues when tracking moving targets with video. Essentially, the component of the target's velocity along a line between it and the camera is ambiguous or unobservable. Therefore, other constraints or assumptions are necessary before target tracking from monocular vision is possible. One possible solution constrains the target motion to a 2D ground plane. After taking video images of the local terrain and targets from a UAV, the authors of [19] reconstruct the camera poses based upon projective homography matrices developed from the camera images and a geo-referenced satellite image. The pose information for each picture provides a means to determine a latitude and longitude position measurement of the target. While novel in some aspects, this method is limited to flat ground planes, and when a geo-referenced image is unavailable, the estimator performance degrades rapidly. Other works appear content on using multiple coordinated UAVs for tracking ground targets. Much of the work outlined in [13] is based upon the realization that inexpensive UAVs are often instrumented with low cost sensors due to the high loss rate associated with autonomous flight and hostile target tracking. Therefore, a necessity for success involves the coordinated flight and collaboration of sensor data obtained from multiple UAVs. This includes the use of several UAVs outfitted with monocular cameras. Multiple camera poses obtained at the same instance provide triangulation of the target's position in 3D space, which negates the camera scaling loss generated by any one camera.

This thesis researches another solution to the target tracking problem. Like [19], the target is constrained to follow the ground terrain and is captured in images obtained from a UAV mounted camera. However, most terrain undulates in 3D space and terrain models are seldom available to the accuracy required. Therefore, this thesis borrows some techniques of SLAM to estimate the 3D terrain near the moving target. Unlike moving targets, stationary features are observable using vision from a moving platform. If the ground features can be accurately mapped, then terrain information, combined with camera states and image measurements, provides an alternative method to determine the 3D target position measurement vector.

The SLAM problem has received substantial attention in recent years. A wide array of sensors and configurations of sensors have been studied including binocular vision, laser scanners, radar, and ultrasonic sensors. Currently, SLAM techniques are applied to monocular vision systems in [2], [3], and [12]. In [2], the authors assume that an air vehicle maintains a constant altitude and that all the features are located in a flat plane. With these assumptions, they can reconstruct the 2D location of the aircraft and a 2D map of the features. In another work involving a UAV with monocular vision, [3], the researchers utilize artificially placed features with a known size. In this case, the range to the fiducials is estimated directly from the image, allowing a 3D reconstruction. Unfortunately, both [2] and [3] have synthetically modified the problem to overcome the image scaling difficulties. In order to find 3D monocular SLAM implementations, more literature searches were required. The research conducted by Andrew Davison et al. explores MonoSLAM techniques as they apply to a system defined by a single camera with no other external sensors [12]. Since their camera does not receive absolute position from a GPS sensor, the estimation filter is initialized with *a priori* knowledge of a few feature locations.

Provided with this initial scale, the filter is able to systematically generate the 3D location of the camera as it varies with time and a map of features locations in 3D space.

1.2 Research Objectives

This thesis seeks to accomplish two major tasks. The first is to develop a filter that will estimate the navigation states of a UAV and the 3D terrain feature locations defining its environment. The model developed must accurately incorporate typical UAV inertial measurement unit (IMU) input data and GPS and camera image output data in a recursive state estimator. This research differs from [2] and [3] by not limiting the environment to a 2D plane and considers features with unknown size. Since GPS is available, scale will not have to defined as in [12].

The next objective is to develop a novel technique which incorporates information obtained from the aircraft and terrain estimator for use in tracking a ground vehicle. This requires redefinition of the observation model to cope with the unobservability issues present with 3D target tracking from a single camera. The solution must also adhere to statistical theory present within filter derivations.

1.3 Estimator Design

The estimator design consists of two loosely coupled (LC) filters: one for SLAM and one for the target. In filter terminology, LC filters obtain input and/or output measurements that are estimates from other filters. Many GPS aided inertial navigation solution (GPS/INS) filters, commonly used for aircraft navigation, are loosely coupled. In these filters, the output GPS position and velocity measurements are actually estimates determined from satellite constellation and pseudo range measurements. In contrast, tightly coupled aircraft navigation filters use the actual satellite information as direct output measurements. In regards to the target tracking estimator setup, LC refers only to separation of aircraft and feature states from target states, although the SLAM filter is truly loosely coupled with respect the GPS position and velocity measurements.

In the first filter, GPS is used to overcome the scale issue and obtain a 3D SLAM estimate of both the UAV's navigation states and the terrain feature locations. The results also apply when the aircraft and feature states are estimated in separate LC estimators, which may be necessary for certain applications. Terrain features that were observed, but have gone out of view, are discarded in order to maintain reasonable computational loads. Thus the map is only maintained in the vicinity of the target. An unscented Kalman filter generates the estimated states and the associated state error covariance matrix for the system. A Feature Manager handles updating the state vector and covariance matrices as required by the addition and deletion of feature states.

The moving target's states are housed within another state estimator that incorporates the estimates from the first filter. An unscented transformation (UT) is employed to propagate the means and uncertainties of the SLAM states and camera measurements into a 3D measurement vector for the target with an estimated covariance. The 3D target position measurement allows both the position and velocity of the target to be estimated where the accuracy of this tracking method is directly correlated with the accuracy of the SLAM estimator.

1.4 Overview

A brief overview of this thesis is given here. Chapter 2 explains the experimental setup used for data collection. This includes describing the airframe and the various electronic hardware components, including sensors, used by a research UAV named the ECat. Chapter 3 details the state equation development for the experimental setup. Chapter 4 reviews background information and important assumptions regarding the Kalman filter family of estimators. A brief heuristic explanation of the observability issues encountered in monocular vision approaches follows in Chapter 5. Chapter 6 details the proposed solution, which utilizes the UT and loosely coupled estimators. This is followed with simulation results in Chapter 7 that illustrate the viability of the theoretical development. Chapter 8 outlines the development and analysis of three different feature initialization techniques. Experimental terrain mapping results are given in Chapter 9 followed by the conclusions and recommendations of Chapter 10.

Chapter 2: Hardware Setup

The experimental data presented in this thesis is collected using the ECat UAV from the Autonomous Vehicle Systems (AVS) Lab at KSU. The research focuses on target tracking using lightweight, inexpensive instruments on a UAV. The hardware on the ECat is typical of what might be expected on such a system. It is outfitted with a digital camera, data collection computer, and an autopilot with the typical set of sensors that are included in inexpensive GPS/INS and air data systems. However, the camera in the ECat is not gimbaled as might be in UAV systems targeted in this research.

2.1 Unmanned Aerial Vehicle Airframe

A SIG Kadet Senior ARF hobbyist kit is used as the basic ECat UAV airframe. The high wing design and approximate 2m wingspan provide stability and adequate lift. Thrust is generated by a Hacker C50 brushless motor with a 16x10" CAM carbon composite folding propeller. Hitec HS-81 servos actuate the elevator, rudder, aileron, and flap control surfaces. The ECat airframe is shown in Fig 2-1.

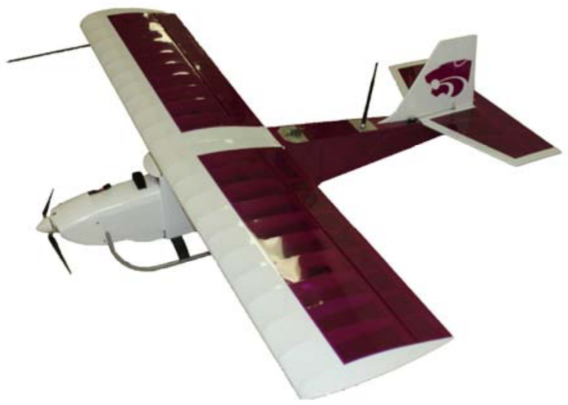


Figure 2-1: ECat UAV

A few modifications to the basic airframe are necessary for hardware accommodation and mission specific requirements. A larger lightweight carbon composite payload bay was designed and inserted in place of the original balsa wood compartment, which relied on several cross-members for structural support. Another noticeable change includes the addition of skid style landing gear, allowing grass prairie landings. Finally, a balsa wood autopilot and camera mount resides in the fore compartment between the firewall and the payload bay bulkhead. This mount is designed to keep the camera and IMU axes orthogonal as is assumed in model development.

2.2 Piccolo II Autopilot

The Piccolo II autopilot, purchased from Cloud Cap Technology, Inc. (CCT), provides the functionality required for the ECat UAV navigation and control applications. With a mass of 233 grams and a 12.2cm x 6.1cm x 3.8cm size, the autopilot avionics unit, pictured in Fig. 2-2, readily mounts within most UAV airframes.

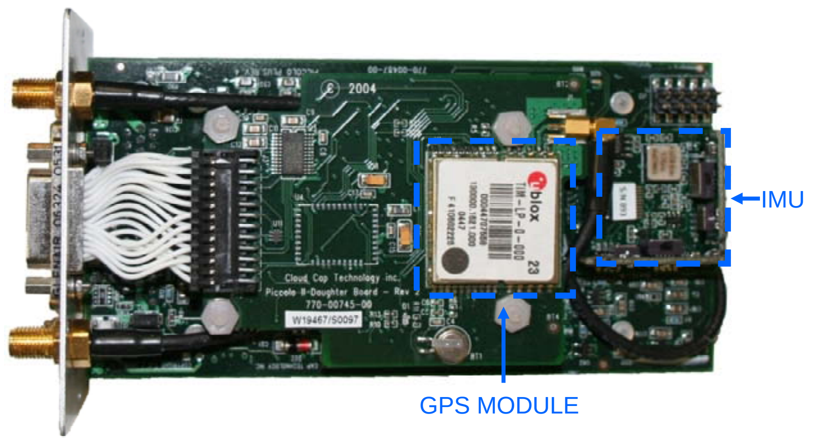


Figure 2-2: Piccolo II Autopilot Hardware

A Motorola MPC555 processor, executing at 40 MHz, provides computation and communication with five RS232 payload ports, up to ten PWM servo channels, two CAN ports, six GPIO pins, and four analog input pins. These peripheral devices are interfaced via a 44 pin D sub connector and a high density 25 pin microdot connector. The daughter cards are comprised of several daughter boards including a MHX-910 Datalink Radio chipset, a Motorola M12 GPS module, an IMU, and dual ported mpxv50045 dynamic pressure and mpx4115a static pressure sensors. The radio link allows the streaming of data to and from a ground station unit that provides a networking interface between multiple avionics units and CCT's operator interface (OI) software. The OI, which executes on a personal computer, displays telemetry updates and also enables the dynamic changing of commands, gains, and flight plans. The IMU, pre-calibrated by CCT, delivers three axis gyro and accelerometer readings to the processor over a serial port. The gyros measure angular rates up to 5.2 radians per second, and the accelerometers record up to 10G accelerations. The Motorola M12 GPS unit generates an estimate of the position and velocity of the Synergy Systems AR-05 antenna. The Piccolo also supports DGPS corrections received from the ground station. The pitot and static ports retrieve air data information vital for true air speed estimation.

CCT also distributes free aircraft simulator and Flight Gear graphical display software as part of their development package. This allows hardware in-the-loop and software in-the-loop laboratory testing necessary for the extensive Piccolo II source code modifications required for this research.

2.3 Marlin Digital Camera

High rate images are captured using a Marlin model MF-201C digital camera from Allied Vision Technologies pictured in Fig. 2-3. With a mass of less than 120 grams and a 5.8cm x 4.4 cm x 2.9 cm dimension (without the lens), this camera provides a practical solution for UAV vision systems. The camera is triggered externally through a HiRose plug connector input pin. The images, up to two Mega pixel resolution, are accessible through a 400 Mb/s Firewire A bus. The lens is a Kowa with an 8mm nominal focal length, which generates an approximate 30° x 39° field of view (FOV).

Figure 2-3: Marlin Digital Camera

2.4 PC104 Stack

A PC104 stack from Kontron collects and stores images and external sensor data for post processing. This 700 MHz Pentium 3 data collection computer was chosen for its small form factor, minimal weight, processing capabilities, and data storage devices. A basic PC104 stack contains a motherboard, flash to IDE converter board, and power supply. Two additional boards were purchased from Advanced Digital Logic and installed on the stack to provide a communication interface to avionics' processor and the digital camera. These include a controller area network (CAN) board and a IEEE 1394 Firewire A board, which both utilize the PC104+ Bus. An additional 2G of removable flash memory is accessible through the USB port. The complete PC104 stack is shown in Fig. 2-4.

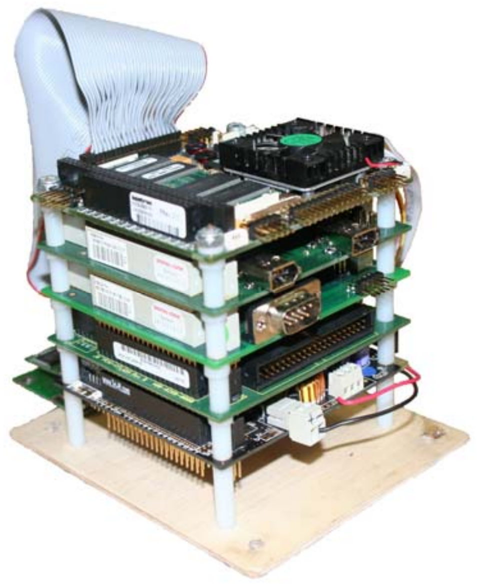


Figure 2-4: PC104 Stack

A Gentoo distribution of the Linux 2.6.22 kernel and operating system provides a versatile programmable interface to the PC104 stack hardware components. Threaded software services the 6.2 Peak Linux CAN driver and the Firewire Linux kernel driver interrupts. The Piccolo avionics' data are collected, parsed, and stored in a text file located on the external memory drive along with the compressed images from the digital camera. The Marlin camera software library is also stored on the PC104 stack and allows camera mode changes and programmable adjustments.

2.5 Target Tracking Hardware Schematic

Fig. 2-5 illustrates the relationships between all of the ECat's hardware components. An 8000 mah LiPo Thunder Power battery provides a nominal 18.5V to the system. Both 12V and 5V BECs regulate voltage input into the Piccolo avionics unit. The Hacker motor controller, PC104 Stack, and voltage divider also have power supplied directly from the battery. The high resistance voltage divider scales the battery voltage from 0-18.5V to 0-5V for analog input battery voltage monitoring. The 900 MHz antenna, GPS antenna, and pitot/static tube mate directly into their respective ports and connectors located on the front panel of the avionics' case. The autopilot supplies PWM signals to actuate the rudder, elevator, aileron, flaps, and motor controller. A CAN 2.0 B bus is used to ferry relevant sensor data from the Piccolo to the PC104 Stack at 20 Hz (autopilot control law rate). CAN is used for communication instead of serial communication for its higher bandwidth capability (up to 1 Mega baud) and its cyclic redundancy checking. The autopilot is also responsible for triggering the camera when new GPS data is available. This is accomplished at a rate of approximately 4Hz with one of the discrete output pins. The PC104 Stack powers the camera and retrieves the triggered images through the Firewire A cable.

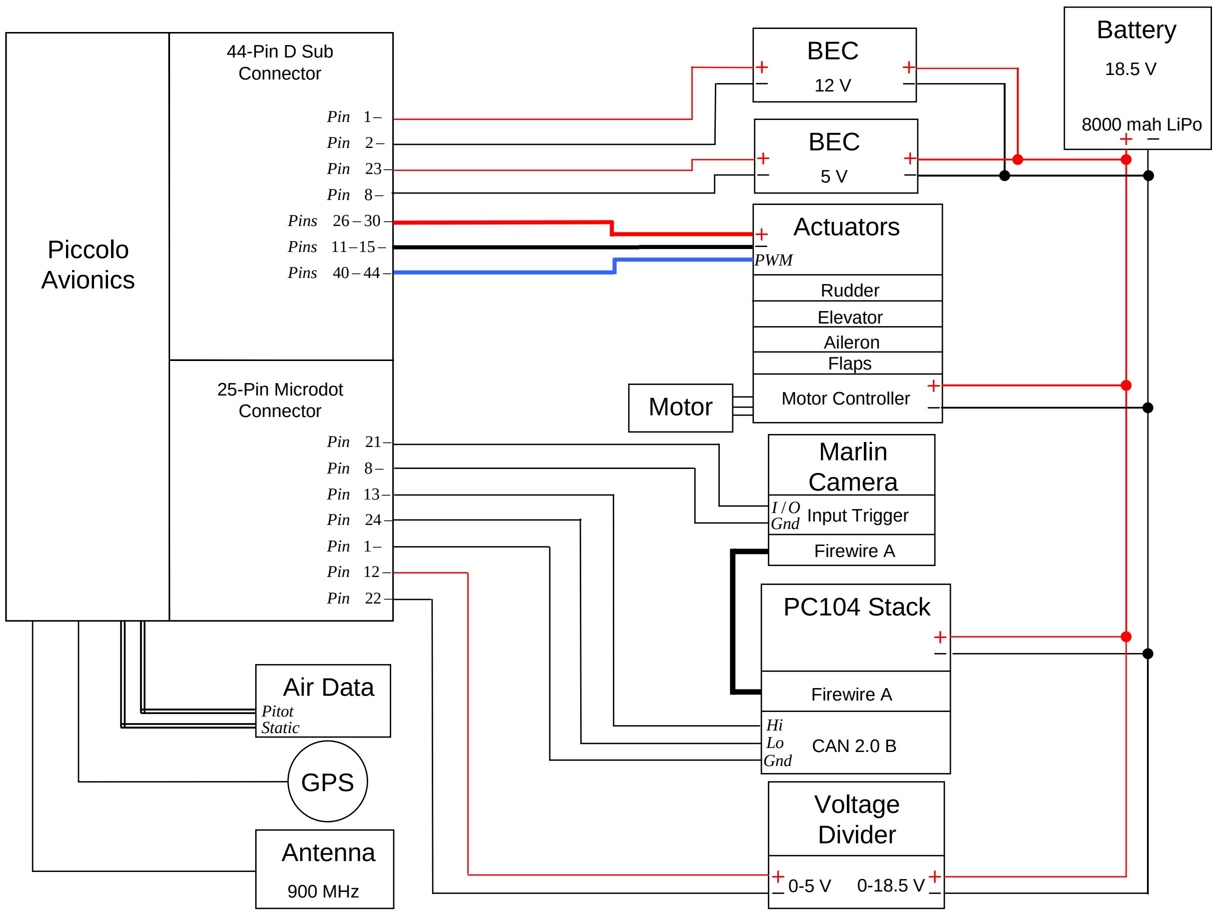


Figure 2-5: Hardware Component Wiring Schematic

Chapter 3: Model Development

An integral step in simulation and estimator development includes defining a set of realistic assumptions that accurately describe the actual hardware. In this thesis, the setup consists of modeling the relatively small ECat UAV flying at low velocity and altitude over varying terrain. Sensor data are available from an IMU, a GPS module, and a monocular digital camera.

3.1 Assumptions

The following list details important assumptions used for model development.

- The aircraft's initial position, or base station, defines the location of the inertial North East Down (NED) frame, $\{i\}$. This assumption is valid for highly maneuverable low velocity aircraft flying over a small area.
- The gravity vector will be constant during the flight. Its orientation will be aligned with the "down" axis of $\{i\}$.
- The three angular rate sensors and three accelerometers within the IMU are located at the center of mass of the aircraft and are aligned with the axes of the body fixed reference frame, $\{b\}$. The GPS antenna is near the center of mass.
- The camera is located at the center of mass pointing down along the z axis of the body frame. Therefore, the camera frame is coincident with the body frame.
- The intrinsic parameters of the camera are known including the focal length, image center, skew coefficient, and radial and tangential distortions.
- A scale invariant distinctive feature based extraction algorithm exists for locating features within discrete images and for specifying the corresponding image coordinate locations from overlapping images.
- Targets are distinguishable from stationary features in images. However, no other properties about the target are known.
- The targets are limited to ground targets that must remain in contact with local ground terrain.

3.2 State Equations

As is typical for GPS aided inertial navigation solutions, kinematic equations rather than kinetic equations are used. This allows for estimation of the aircraft and target states without knowledge of the intrinsic parameters of the aircraft and target. Therefore, the resulting state estimation technique is less dependent on the specific application. Kinematic relationships generally tend to decrease the number of states necessary for estimation while providing a quantifiable means for determining output and process noise.

For the development, three sets of kinematic equations are required: one for the UAV, one for the features, and one for the target. The general form for these nonlinear state equations is given by

$$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u}, \vec{w}) \tag{3.1}$$

$$\vec{z} = \vec{h}(\vec{x}, \vec{v}) \tag{3.2}$$

where $\vec{x}$ is state vector, $\vec{u}$ is the input vector, $\vec{w}$ is the zero mean white input noise vector, $\vec{v}$ is the zero mean white output noise vector, and $\vec{z}$ is the output vector. In the following discussion, (3.1) will be referred to as the dynamic model as it defines the relationship between the derivative of the state vector and the nonlinear vector function, $\vec{f}(\cdot)$, of the states, inputs, and process noise. Equation (3.2) will be referred to as the observation or output model where the nonlinear output vector function, $\vec{h}(\cdot)$, defines the combination of states and output noise that form the output vector.

3.2.1 Aircraft Nonlinear State Equations

The definitions for the aircraft state vector, input vector, and output vector are shown below. The “a” subscript denotes aircraft states.

$$\vec{x}_a \equiv \begin{bmatrix} \vec{p}_{a\{i\}} \\ \dot{\vec{p}}_{a\{i\}} \\ \vec{q} \end{bmatrix} \tag{3.3}$$

$$\vec{u}_a \equiv \begin{bmatrix} \vec{a}_{\{b\}} \\ \vec{\omega}_{\{b\}} \end{bmatrix} \tag{3.4}$$

$$\vec{z}_a \equiv \begin{bmatrix} \vec{p}_{a\{i\}} \\ \dot{\vec{p}}_{a\{i\}} \end{bmatrix} \tag{3.5}$$

The state vector, $\vec{x}_a$, consists of the UAV's inertial position vector, $\vec{p}_{a\{i\}}$, inertial velocity vector, $\dot{\vec{p}}_{a\{i\}}$, and the quaternion vector, $\vec{q}$, where

$$\vec{q} \equiv [q_0 \quad q_1 \quad q_2 \quad q_3]^T . \tag{3.6}$$

The input vector, $\vec{u}_a$, contains the body axes accelerations, $\vec{a}_{\{b\}}$, and angular rates, $\vec{\omega}_{\{b\}}$. The output, $\vec{z}_a$, contains the inertial position vector, $\vec{p}_{a\{i\}}$, and inertial velocity vector, $\dot{\vec{p}}_{a\{i\}}$.

Quaternions are used to represent the orientation between the inertial and body fixed frames instead of their Euler angle counterparts. This avoids the singularity condition apparent with Euler angle attitude representations. A detailed description of the quaternion vector elements, quaternion to Euler angle conversions, and the quaternion inertial to body frame rotation matrix is located in Appendix A.

3.2.1.1 Aircraft Dynamic Equations

The aircraft dynamic equations are given by $\dot{\vec{x}}_a$, the time derivative of the state vector. The time derivative of position within the inertial frame, $\dot{\vec{p}}_{a\{i\}}$, is simply the inertial velocity vector.

$$\frac{d}{dt}(\vec{p}_{a\{i\}}) = \dot{\vec{p}}_{a\{i\}} \tag{3.7}$$

The time rate of change of the velocity vector, $\ddot{\vec{p}}_{a(i)}$, represents the true inertial kinematic accelerations. However, the accelerometers located on the body frame will measure the true kinematic accelerations, $\vec{a}_{\{b\}}$, corrupted with components of zero mean white noise, $\vec{w}_{a\{b\}}$, and the body frame representation of the constant magnitude gravity vector, g. Hence, the acceleration measurement is given as

$$\vec{a}_{m\{b\}} = \vec{a}_{\{b\}} + \vec{w}_{a\{b\}} - \mathbf{R}_{\{ib\}}[0 \quad 0 \quad g]^T \tag{3.8}$$

where $\mathbf{R}_{\{ib\}}$ is the body frame to inertial frame rotation matrix. The sign of the zero mean noise term is arbitrary. In (3.8) and subsequent developments, the noise terms are added to the true states strictly for notational consistency. The true inertial accelerations, $\vec{a}_{\{i\}}$, are obtained by solving (3.8) for the true body frame accelerations and rotating this vector into the inertial frame.

$$\ddot{\vec{p}}_{a\{i\}} = \vec{a}_{\{i\}} = \mathbf{R}_{\{ib\}}\vec{a}_{\{b\}} \tag{3.9}$$

$$\ddot{\vec{p}}_{a\{i\}} = \mathbf{R}_{\{ib\}}\left(\vec{a}_{m\{b\}} - \vec{w}_{a\{b\}}\right) + \begin{bmatrix}0 & 0 & g\end{bmatrix}^T \tag{3.10}$$

The other IMU input contains the angular velocity vector measurements from the rate gyros, $\vec{\omega}_{m\{b\}}$. In accordance with the accelerometers, this measurement contains the true angular velocities and zero mean white noise, $\vec{w}_{m\{b\}}$.

$$\vec{\omega}_{m\{b\}} = \vec{\omega}_{\{b\}} + \vec{w}_{\omega\{b\}} \tag{3.11}$$

The relationship between the time derivatives of the quaternion states, whose elements are functions of the rotation matrix, and the gyro measurements is known as the "strapdown" equations [6]

$$\dot{\vec{q}} = \frac{1}{2}\mathbf{\Omega}(q)(\vec{\omega}_{\{b\}}) \tag{3.12}$$

where

$$\mathbf{\Omega}(q) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}. \tag{3.13}$$

Solving (3.11) for the true angular velocity vector and substituting into (3.12) results in

$$\dot{\vec{q}} = \frac{1}{2}\mathbf{\Omega}(q)(\vec{\omega}_{m\{b\}} - \vec{w}_{\omega\{b\}}). \tag{3.14}$$

The concatenation of equations (3.7), (3.10), and (3.14) results in the nonlinear aircraft dynamic equation vector.

$$\dot{\vec{x}}_a = \begin{bmatrix} \dot{\vec{p}}_{a\{i\}} \\ \ddot{\vec{p}}_{a\{i\}} \\ \dot{\vec{q}} \end{bmatrix} = \begin{bmatrix} \dot{\vec{p}}_{a\{i\}} \\ \mathbf{R}_{\{ib\}}\left(\vec{a}_{m\{b\}} - \vec{w}_{a\{b\}}\right) + \begin{bmatrix}0 & 0 & g\end{bmatrix}^T \\ \frac{1}{2}\mathbf{\Omega}(q)\left(\vec{\omega}_{m\{b\}} - \vec{w}_{\omega\{b\}}\right) \end{bmatrix} \tag{3.15}$$

3.2.1.2 *Aircraft Observation Equations*

The output measurements given by the GPS includes estimates of both inertial position, $\vec{p}_{am\{i\}}$, and inertial velocity, $\dot{\vec{p}}_{am\{i\}}$. Both of these measurement vectors contain the true inertial position and velocity and their corresponding noise components, $\vec{v}_{p_a\{i\}}$ and $\vec{v}_{\dot{p}_a\{i\}}$.

$$\vec{z}_a = \begin{bmatrix} \vec{p}_{a\{i\}} \\ \dot{\vec{p}}_{a\{i\}} \end{bmatrix} = \begin{bmatrix} \vec{p}_{am\{i\}} \\ \dot{\vec{p}}_{am\{i\}} \end{bmatrix} - \begin{bmatrix} \vec{v}_{p_a\{i\}} \\ \vec{v}_{\dot{p}_a\{i\}} \end{bmatrix} \tag{3.16}$$

3.2.2 *Feature Nonlinear State Equations*

The following definitions list the state vector, input vector, and output vector associated with the features. The "f" subscript denotes feature states.

$$\vec{x}_f \equiv \begin{bmatrix} \vec{p}^{\,1}_{f\{i\}} \\ \vdots \\ \vec{p}^{\,N_f}_{f\{i\}} \end{bmatrix} \tag{3.17}$$

$$\vec{u}_f \equiv \begin{bmatrix} \vec{0} \\ \vdots \\ \vec{0} \end{bmatrix} \tag{3.18}$$

$$\vec{z}_f \equiv \begin{bmatrix} \vec{i}^{\,1}_{f\{b\}} \\ \vdots \\ \vec{i}^{\,N_f}_{f\{b\}} \end{bmatrix} \tag{3.19}$$

The feature state vector, $\vec{x}_f$, in (3.17) contains the feature vector locations, $\vec{p}_{f\{i\}}$, relative to the NED inertial frame. The number of estimated features may vary with time as new features come into observation and other features go out of observation. The superscripts are used to identify the various features where N_f denotes the total number of features comprising the state vector at any given time. The variable size input vector, $\vec{u}_f$, in (3.18) is all zero vectors as the features are stationary. The output vector, $\vec{z}_f$, contains the image space vector, $\vec{i}_{f\{b\}}$, for each feature. Each $\vec{i}_{f\{b\}}$ consists of two elements which define the measured x and y components of a single image space feature representation relative to $\{b\}$.

3.2.2.1 Feature Dynamic Equations

Features are assumed stationary, which implies the time derivative of the feature state vector, $\dot{\vec{x}}_f$, is zero.

$$\dot{\vec{x}}_f = \dot{\vec{p}}_{f\{i\}} = \vec{0} \tag{3.20}$$

3.2.2.2 Feature Observation Equations

A monocular camera installed along the body axes of the aircraft captures images at periodic intervals. Unfortunately these images do not provide a depth measurement to the feature located in 3D space, but instead capture its image space coordinates. Therefore, the feature observation equations need to determine the relationships between the aircraft position and orientation, the intrinsic camera parameters, and the feature's position that result in its image coordinate representation. Fig. 3-1 depicts the relationships between the inertial frame, the camera frame, and a feature viewed through a pinhole camera model.

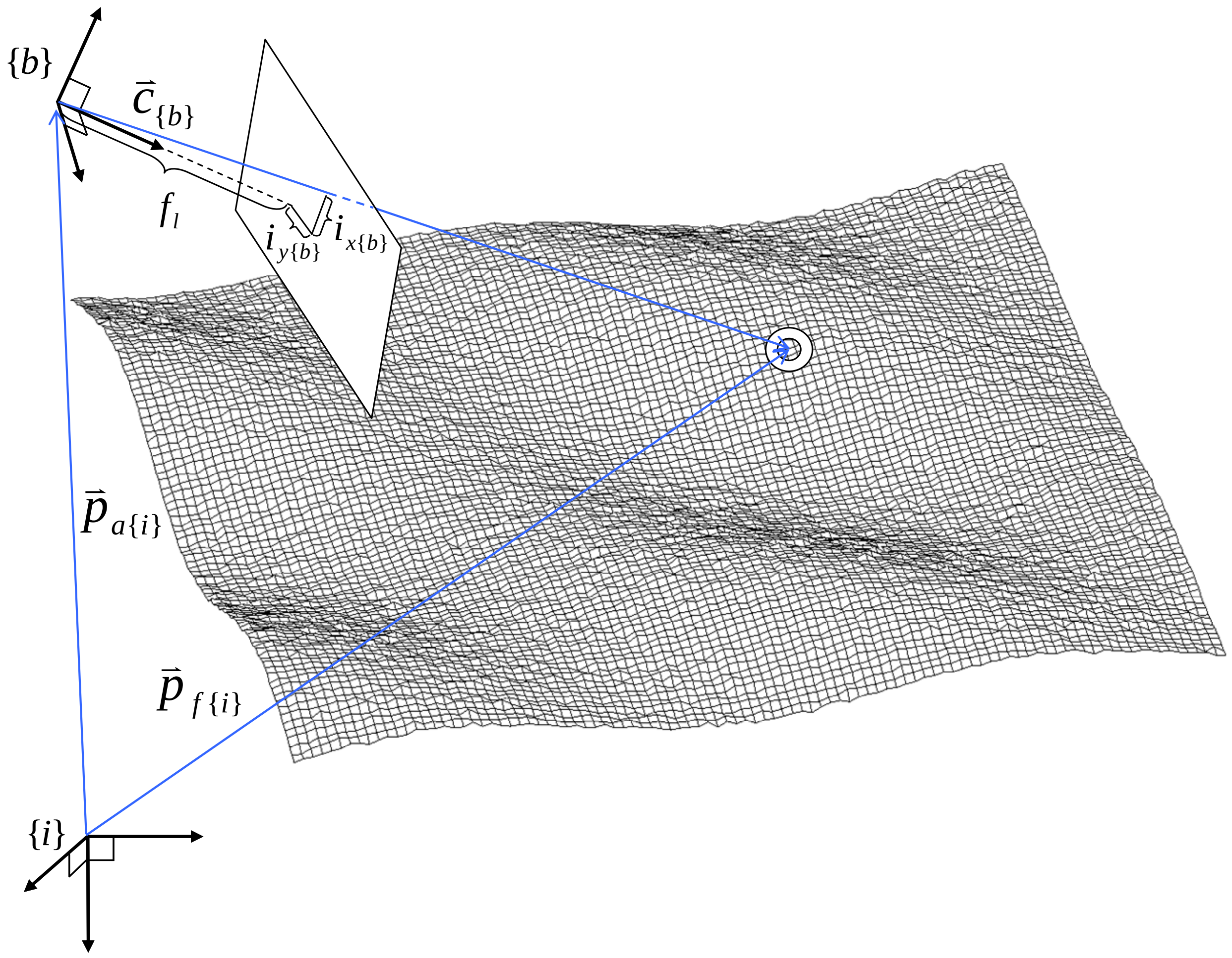


Figure 3-1: Feature Observation Development Assuming Pinhole Camera Model

The vector, $\vec{c}_{\{b\}}$, from the camera frame to the feature, is given by the following equation

$$\vec{c}_{\{b\}} = \mathbf{R}_{\{bi\}}(\vec{p}_{f\{i\}} - \vec{p}_{a\{i\}}) \tag{3.21}$$

where the rotation matrix, $\mathbf{R}_{\{bi\}}$, transforms the inertial frame representations of the feature and aircraft into the body frame. The image space feature coordinates are realized when the z component of $\vec{c}_{\{b\}}$ is equal to the focal length, f_l, of the camera. Therefore, the actual images coordinates $i_{x\{b\}}$ and $i_{y\{b\}}$ of a feature are computed as follows.

$$\vec{i}_{f\{b\}} = \begin{bmatrix} i_{x\{b\}} \\ i_{y\{b\}} \end{bmatrix} = \frac{f_l}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\vec{c}_{\{b\}}} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\vec{c}_{\{b\}} \tag{3.22}$$

The measured image space coordinate vectors, $\vec{i}_{fm\{b\}}$, consist of the true image coordinate vector, $\vec{i}_{f\{b\}}$, plus a noise vector, $\vec{v}_{i_f\{b\}}$, due to camera calibration and feature extraction errors. Therefore, the true measured image space coordinates are given by

$$\vec{z}_f = \begin{bmatrix} \vec{i}^{\,1}_{f\{b\}} \\ \vdots \\ \vec{i}^{\,N_f}_{f\{b\}} \end{bmatrix} = \begin{bmatrix} \vec{i}^{\,1}_{fm\{b\}} \\ \vdots \\ \vec{i}^{\,N_f}_{fm\{b\}} \end{bmatrix} - \begin{bmatrix} \vec{v}^{\,1}_{i_f\{b\}} \\ \vdots \\ \vec{v}^{\,N_f}_{i_f\{b\}} \end{bmatrix}. \tag{3.23}$$

An implication of the feature observation model exists when the z component of $\bar{c}_{\{b\}}$ becomes negative. This negative value switches the signs of the pixel coordinates. The x-y plane of the body axes defines this "singularity" region. In fact, if the z component of $\bar{c}_{\{b\}}$ approaches zero, the pixel coordinates $i_{x\{b\}}$ and $i_{y\{b\}}$ become infinite. In reality, this is never a concern as the field of view of the camera is less than 180 degrees. However, in filter implementations, this is possible as several of the estimated states, especially orientation, used in (3.21) and (3.22) contain a high degree of uncertainty. In practice, this issue is addressed by eliminating image updates during periods of poor orientation estimates, which usually occur during filter initialization.

3.2.3 Target Nonlinear State Equations

Like the aircraft, the target state equations consist of kinematic relationships. However, unlike the aircraft model, no direct acceleration measurements are available from the target. One commonly used motion model assumes the velocity of the target is a random walk with acceleration modeled using zero mean Gaussian white noise. For this technique, a larger standard deviation in acceleration noise corresponds to a target that is more maneuverable. This results in greater uncertainty in the target states predicted by the model, thereby weighting the observation more heavily than the prediction in the state estimation process. In this thesis, the target velocity is modeled as a random walk with unchanging noise statistics and dynamic model. For highly maneuverable targets, the Interacting Multiple Model (IMM) estimator can provide improved accuracy at the expense of increased computational complexity. The interested reader should consult [1], [7], [8], and [14]. The target tracking concepts developed in this thesis pertain equally to both the IMM target estimator and the basic single model estimator used for the remainder of this development.

The following equations represent the target state vector, input vector, and output vector. The "t" subscript denotes target states.

$$\vec{x}_t \equiv \begin{bmatrix} \vec{p}_{t\{i\}} \\ \dot{\vec{p}}_{t\{i\}} \end{bmatrix} \tag{3.24}$$

$$\vec{u}_t \equiv \vec{w}_{t\{i\}} \tag{3.25}$$

$$\vec{z}_t \equiv \vec{i}_{t\{b\}} \tag{3.26}$$

In (3.24), the target state vector, $\vec{x}_t$, contains the target position vector, $\vec{p}_{t\{i\}}$, and velocity vector, $\dot{\vec{p}}_{t\{i\}}$, relative to the inertial frame. Since the velocity is modeled using a random walk, the input vector for the target model, $\vec{u}_t$, is a noise term, $\vec{w}_{t\{i\}}$. The target output vector, $\vec{z}_t$, consists of the camera sensor vector, $\vec{i}_{t\{b\}}$, which contains the two image space coordinates of the target.

3.2.3.1 Target Dynamic Equations

The time derivative of the target state vector, $\dot{\vec{x}}_t$, is a concatenation of the velocity and acceleration terms.

$$\frac{d}{dt}(\vec{p}_{t\{i\}}) = \dot{\vec{p}}_{t\{i\}} \tag{3.27}$$

$$\ddot{\vec{p}}_{t\{i\}} = \vec{w}_{t\{i\}} \tag{3.28}$$

Therefore, $\dot{\vec{x}}_t$ is written compactly as

$$\dot{\vec{x}}_t = \begin{bmatrix} \dot{\vec{p}}_{t\{i\}} \\ \ddot{\vec{p}}_{t\{i\}} \end{bmatrix} = \begin{bmatrix} \dot{\vec{p}}_{t\{i\}} \\ \vec{w}_{t\{i\}} \end{bmatrix}. \tag{3.29}$$

3.2.3.2 Target Observation Equations

The target observation model is very similar to the feature observation model since the camera image provides the only output measurement for the target. The image space coordinate vector, $\vec{i}_{t\{b\}}$, is given by (3.22) when $\vec{p}_{f\{i\}}$ in (3.21) is replaced with $\vec{p}_{t\{i\}}$.

$$\vec{c}_{\{b\}} = \mathbf{R}_{\{bi\}}(\vec{p}_{t\{i\}} - \vec{p}_{a\{i\}}) \tag{3.30}$$

$$\vec{i}_{t\{b\}} = \begin{bmatrix} i_{x\{b\}} \\ i_{y\{b\}} \end{bmatrix} = \frac{f_l}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \vec{c}_{\{b\}}} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \vec{c}_{\{b\}} \tag{3.31}$$

The target measurement pixel measurement vector, $\vec{i}_{tm\{b\}}$, consists of the true image space vector, $\vec{i}_{t\{b\}}$, positively perturbed by a noise vector, $\vec{v}_{i_t\{b\}}$, for similar reasons as described for the feature observation equations.

$$\vec{z}_t = \vec{i}_{t\{b\}} = \vec{i}_{tm\{b\}} - \vec{v}_{i_t\{b\}} \tag{3.32}$$

Chapter 4: Observer Research

The standard procedure for estimating the states of a system involves implementing an observer. Observers utilize dynamic/observation models, input, and error feedback during state estimation. The robustness of the observer is highly correlated with how accurately the true system is modeled. In theory, complete knowledge of a linear system model allows for any desired pole placement for the observer. This is possible since the observer is implemented in software, which negates any physical limitations. In actuality, unmodeled dynamics and noisy input and output data limit the achievable bandwidth of the state estimator. The state equations developed in the previous sections are no exception. The input and output sensors are all assumed to include zero mean, white noise components. The Kalman filter family of estimators considers the noise statistics, current estimate uncertainty, and state equations during state estimation. Background information regarding the Kalman filter (KF), extended Kalman filter (EKF), and unscented Kalman filter (UKF) is presented in this chapter.

A typical discrete time observer setup and its relationships with a linear time invariant system are shown in Fig. 4-1. The continuous time linear state equation model for this system is given by

$$\dot{\vec{x}} = \mathbf{F}\vec{x} + \mathbf{G}\vec{u} \tag{4.1}$$

$$\vec{z} = \mathbf{H}\vec{x} \tag{4.2}$$

where $\vec{x}$ is the state vector, $\vec{u}$ is the input vector, and $\vec{z}$ is the output vector. This system is described in discrete time by

$$\vec{x}_k = \mathbf{\Phi}\vec{x}_{k-1} + \mathbf{\Gamma}\vec{u}_{k-1} \tag{4.3}$$

$$\vec{z}_k = \mathbf{H}\vec{x}_k \, . \tag{4.4}$$

$\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are often referred to as the state transition model and control input model, respectively. Their actual values may vary slightly depending upon the discretization method used and the integration time-step.

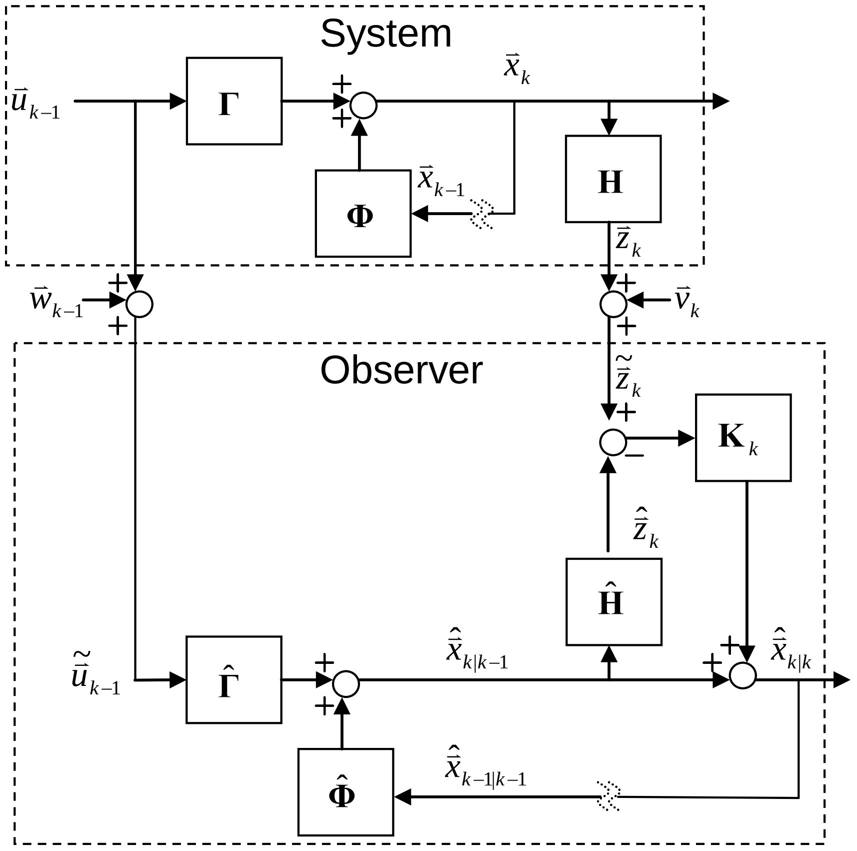


Figure 4-1: Discrete Time Linear System and Observer Relationships

The observer block contains additional complexities. The observer receives input, $\tilde{\vec{u}}_{k-1}$, and output, $\tilde{\vec{z}}_k$, measurements that are corrupted with their respective noise components, $\vec{w}_{k-1}$ and $\vec{v}_k$. The major difference between the observer and the system models is the correction step. The error between the observer output, $\hat{\vec{z}}_k$, and the actual system measured output, $\tilde{\vec{z}}_k$, is multiplied by a feedback gain and used to correct the predicted state estimate, $\hat{\vec{x}}_{k|k-1}$.

From another perspective, the observer illustrated in Fig. 4-1 is estimating the system defined by (4.1) and (4.2), which can be rewritten as

$$\vec{x}_k = \mathbf{\Phi}\vec{x}_{k-1} + \mathbf{\Gamma}\tilde{\vec{u}}_{k-1} - \mathbf{\Gamma}\vec{w}_{k-1} \tag{4.5}$$

$$\tilde{\vec{z}}_k = \mathbf{H}\vec{x}_k + \vec{v}_k \tag{4.6}$$

where

$$\tilde{\vec{u}}_{k-1} = \vec{u}_{k-1} + \vec{w}_{k-1} \tag{4.7}$$

$$\tilde{\vec{z}}_k = \vec{z}_k + \vec{v}_k\,. \tag{4.8}$$

These equations take the standard form for the discrete time KF when the process noise, $\vec{w}_{k-1}$, originates from the input source. Various forms of the KF equations exist, including those that take into consideration plant modeling and/or plant disturbance errors. While only input noise disturbances are considered here due to the structure of the model equations previously developed, the concepts are readily extendable to the other cases.

4.1 Kalman Filter

The KF can be interpreted as a recursive minimum least squares estimator, which optimally determines the states of a linear system with input, output, and/or process noise. It is optimal in the sense that it minimizes the trace of the estimated state error covariance matrix. Therefore, the foundation of the filter relies upon accurate modeling of the propagation of the mean and covariance matrix of random variables (RV) through linear transformations. This is accomplished by recalling the linear transformation properties for RVs. Suppose that a vector, $\vec{z}$, is related to a random vector, $\vec{x}$, through the linear transformation, $\mathbf{H}$.

$$\vec{z} = \mathbf{H}\vec{x} \tag{4.9}$$

Taking the expected value, $E\{\cdot\}$, of (4.9) results in an expression for the mean.

$$E\{\vec{z}\} = E\{\mathbf{H}\vec{x}\} \tag{4.10}$$

$$\bar{\vec{z}} = \mathbf{H}\bar{\vec{x}} \tag{4.11}$$

The covariance matrix, $\mathbf{P}^{zz}$, for $\vec{z}$ is found by its definition.

$$\mathbf{P}^{zz} = E\left\{\left(\vec{z} - \bar{\vec{z}}\right)\left(\vec{z} - \bar{\vec{z}}\right)^T\right\}$$

$$\mathbf{P}^{zz} = E\left\{\mathbf{H}\left(\vec{x} - \bar{\vec{x}}\right)\left(\vec{x} - \bar{\vec{x}}\right)^T \mathbf{H}^T\right\} \tag{4.12}$$

This is simply

$$\mathbf{P}^{zz} = \mathbf{H}\mathbf{P}^{xx}\mathbf{H}^T \tag{4.13}$$

where the covariance matrix $\mathbf{P}^{xx}$ is defined by $E\left\{\left(\vec{x} - \bar{\vec{x}}\right)\left(\vec{x} - \bar{\vec{x}}\right)^T\right\}$. These principles are used in the KF to determine the mean and covariance propagation of RVs through linear dynamic and observation models. The Kalman gain, $\mathbf{K}$, is then determined which minimizes the trace of the estimated state error covariance matrix. If the RVs have Gaussian probability density functions

(PDF), then they are fully described by their corresponding means and covariance matrices, and the KF provides the optimal solution. A weaker condition occurs when the input and measurement noises are white and zero mean, but not necessarily Gaussian. In this case, the KF provides the optimal linear solution [15]. The filter accomplishes these tasks in two distinct phases, prediction and correction.

4.1.1 Prediction

During the prediction phase, both the current estimated state mean and error covariance matrix are updated. The new state estimate $\hat{\vec{x}}_{k|k-1}$ is given by

$$\hat{\vec{x}}_{k|k-1} = \mathbf{\Phi}\hat{\vec{x}}_{k-1|k-1} + \mathbf{\Gamma}\tilde{\vec{u}}_{k-1}. \tag{4.14}$$

Essentially, (4.14) represents the time integration of the dynamic equation with known initial conditions and corrupted input, $\tilde{\vec{u}}_{k-1}$. The next step involves calculation of the updated state error covariance matrix. This new covariance matrix , $\mathbf{P}^{xx}_{k|k-1}$, is given by its definition.

$$\mathbf{P}^{xx}_{k|k-1} = E\left\{\left(\hat{\vec{x}}_{k|k-1} - \vec{x}_k\right)\left(\hat{\vec{x}}_{k|k-1} - \vec{x}_k\right)^T\right\} \tag{4.15}$$

Substituting in the dynamic model, (4.3), for $\vec{x}_k$ and $\hat{\vec{x}}_{k|k-1}$ gives

$$\mathbf{P}^{xx}_{k|k-1} = E\left\{\left(\mathbf{\Phi}\left(\hat{\vec{x}}_{k|k-1} - \vec{x}_{k-1}\right) + \mathbf{\Gamma}\left(\vec{w}_{k-1}\right)\right)\left(\mathbf{\Phi}\left(\hat{\vec{x}}_{k|k-1} - \vec{x}_{k-1}\right) + \mathbf{\Gamma}\left(\vec{w}_{k-1}\right)\right)^T\right\} \tag{4.16}$$

where

$$\vec{w}_{k-1} = \tilde{\vec{u}}_{k-1} - \vec{u}_{k-1}. \tag{4.17}$$

Since $\vec{w}_{k-1}$ is uncorrelated with the other terms in (4.16), $\mathbf{P}^{xx}_{k|k-1}$ can be written as

$$\mathbf{P}^{xx}_{k|k-1} = E\left\{\left(\mathbf{\Phi}\left(\hat{\vec{x}}_{k-1|k-1} - \vec{x}_{k-1}\right)\right)\left(\mathbf{\Phi}\left(\hat{\vec{x}}_{k-1|k-1} - \vec{x}_{k-1}\right)\right)^T\right\} + E\left\{\left(\mathbf{\Gamma}\left(\vec{w}_{k-1}\right)\right)\left(\mathbf{\Gamma}\left(\vec{w}_{k-1}\right)\right)^T\right\}. \tag{4.18}$$

Realizing the linear properties of covariance, (4.18) is simply

$$\mathbf{P}^{xx}_{k|k-1} = \mathbf{\Phi}\mathbf{P}^{xx}_{k-1|k-1}\mathbf{\Phi}^T + \mathbf{\Gamma}\mathbf{Q}_{k-1}\mathbf{\Gamma}^T \tag{4.19}$$

where

$$\mathbf{P}^{xx}_{k-1|k-1} = E\left\{\left(\hat{\vec{x}}_{k-1|k-1} - \vec{x}_{k-1}\right)\left(\hat{\vec{x}}_{k-1|k-1} - \vec{x}_{k-1}\right)^T\right\} \tag{4.20}$$

$$\mathbf{Q}_{k-1} = E\left\{\left(\vec{w}_{k-1}\right)\left(\vec{w}_{k-1}\right)^T\right\}. \tag{4.21}$$

This series of prediction steps occurs at the rate of new input data. Note that the development applies to both linear time invariant systems and linear time varying systems, although only time invariant systems are shown here for notation simplicity.

4.1.2 Correction

When an output measurement is available, the estimated state is updated by

$$\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1} + \mathbf{K}_k (\tilde{\vec{z}}_k - \mathbf{H}\hat{\vec{x}}_{k|k-1}) \tag{4.22}$$

where

$$\tilde{\vec{z}}_k - \mathbf{H}\hat{\vec{x}}_{k|k-1} \tag{4.23}$$

is the measurement residual or error. The measurement vector, $\tilde{\vec{z}}_k$, also contains zero mean white noise, $\vec{v}_k$.

$$\tilde{\vec{z}}_k = \mathbf{H}\vec{x}_k + \vec{v}_k \tag{4.24}$$

This correction results in a new posterior estimate covariance matrix, $\mathbf{P}^{xx}_{k|k}$, which is defined by

$$\mathbf{P}^{xx}_{k|k} = E\left\{\left(\hat{\vec{x}}_{k|k} - \vec{x}_k\right)\left(\hat{\vec{x}}_{k|k} - \vec{x}_k\right)^T\right\}. \tag{4.25}$$

Substituting (4.22) and (4.24) into (4.25), expanding the terms, and simplifying the expression using covariance properties results in an equation for $\mathbf{P}^{xx}_{k|k}$

$$\mathbf{P}^{xx}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}^{xx}_{k|k-1}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{4.26}$$

that is a function of $\mathbf{I}$ (Identity Matrix), $\mathbf{K}_k$, $\mathbf{H}$, $\mathbf{P}^{xx}_{k|k-1}$, and $\mathbf{R}_k$, where

$$\mathbf{R}_k = E\left\{\left(\vec{v}_k\right)\left(\vec{v}_k\right)^T\right\}. \tag{4.27}$$

Up to this point, the Kalman gain determination has not been addressed. However, the covariance, or uncertainty, of the updated state estimate is a function of this gain. Therefore, this gain is chosen to minimize the expected error, $E\left\{\left\|\hat{\vec{x}}_{k|k} - \vec{x}_k\right\|\right\}$, which is the trace of $\mathbf{P}^{xx}_{k|k}$. Setting the partial of the trace of $\mathbf{P}^{xx}_{k|k}$ with respect to the Kalman gain equal to zero results in an expression that is solved for the optimal gain.

$$\mathbf{K}_k = \mathbf{P}^{xx}_{k|k-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}^{xx}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k)^{-1} \tag{4.28}$$

The posterior error covariance matrix update is performed by using this gain.

$$\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1} - \mathbf{K}_k\mathbf{H}\mathbf{P}^{xx}_{k|k-1} \tag{4.29}$$

A compact algorithm for the KF is located in Appendix B.

Several intuitive properties result from the Kalman gain and its calculation given in (4.28). The gain is entirely determined by the current state error covariance matrix, the output matrix, and the measurement noise statistics, where the current uncertainty is found with the dynamic model and input noise statistics. In theoretical applications where noise is not considered, the poles of the observer are placed to provide a very quick transient response to disturbances and incorrect initial state estimates. For systems that have noisy input and output measurements, choosing a feedback gain in this manner results in a suboptimal estimate that may even diverge. However, the optimal Kalman gain determines the pole location based upon the current uncertainty. This uncertainty essentially defines the feedback gain, and thus the bandwidth, of the system. An interesting characteristic also arises when the measurement vector is noise free and the output matrix is square. In this case, the gain becomes

$$\mathbf{K}_k = \mathbf{H}^{-1}. \tag{4.30}$$

The use of this gain in (4.22) forces the new current state estimate to be updated in one correction to the actual state, as observed through the observation model, and sets the posterior covariance matrix equal to the zero matrix. In other words, an uncorrupted measurement vector update mitigates any doubt about the uncertainty of the state estimates when the observation matrix is square and nonsingular. Another interesting characteristic happens when the current state error covariance matrix tends toward the zero matrix. When this happens, the current state mean estimates are very likely to be near the actual states. Equation (4.28) then determines a feedback gain that is also near zero since the innovation covariance, $\mathbf{HP}_{k|k-1}^{xx}\mathbf{H}^T + \mathbf{R}_k$, is pre-multiplied by a small magnitude matrix. A null feedback gain emphasizes the fact that the system mean is near the true mean and any noisy measurement should therefore have very little or no effect on the current estimates.

4.2 Extended Kalman Filter

The KF provides an optimal state estimate for linear systems. However, most practical systems, including the developments in this thesis, are defined by nonlinear state equations of the form given in (3.1) and (3.2). The problem of estimating the predicted mean and covariance from nonlinear functions has been addressed by many filtering techniques including the EKF. The following equation shows the first several elements of a Taylor series expansion of a nonlinear function $\vec{h}(\cdot)$ about an operating point defined by the mean of a random variable, $\bar{\vec{x}}$.

$$\vec{z} = \vec{h}(\bar{\vec{x}}) + \left(\left. \frac{\partial(\vec{h}(\vec{x}))}{\partial \vec{x}} \right|_{\vec{x}=\bar{\vec{x}}} \right)(\vec{x} - \bar{\vec{x}}) + \frac{1}{2}(\vec{x} - \bar{\vec{x}})^T \left(\left. \frac{\partial^2(\vec{h}(\vec{x}))}{\partial \vec{x}^2} \right|_{\vec{x}=\bar{\vec{x}}} \right)(\vec{x} - \bar{\vec{x}}) + \cdots \tag{4.31}$$

If the second and higher order terms of the expansion are considered negligible, then $\vec{z}$ is approximated by

$$\vec{z} \approx \vec{h}(\bar{\vec{x}}) + \left(\left. \frac{\partial(\vec{h}(\vec{x}))}{\partial \vec{x}} \right|_{\vec{x}=\bar{\vec{x}}} \right)(\vec{x} - \bar{\vec{x}}). \tag{4.32}$$

With this assumption, the new mean is predicted using the nonlinear function, and the new covariance is found by taking expected value of the outer product of (4.32)

$$\bar{\vec{z}} = \vec{h}(\bar{\vec{x}}) \tag{4.33}$$

$$\mathbf{P}^{zz} = E\left\{(\vec{z} - \bar{\vec{z}})(\vec{z} - \bar{\vec{z}})^T\right\} \tag{4.34}$$

$$\mathbf{P}^{zz} = \left(\left. \frac{\partial(\vec{h}(\vec{x}))}{\partial \vec{x}} \right|_{\vec{x}=\bar{\vec{x}}} \right) \mathbf{P}^{xx} \left(\left. \frac{\partial(\vec{h}(\vec{x}))}{\partial \vec{x}} \right|_{\vec{x}=\bar{\vec{x}}} \right)^T \tag{4.35}$$

where

$$\mathbf{P}^{xx} = E\left\{(\vec{x} - \bar{\vec{x}})(\vec{x} - \bar{\vec{x}})^T\right\}. \tag{4.36}$$

4.2.1 Modifications to Kalman Filter

The KF equations require modifications to employ the mean and covariance estimation techniques of the EKF. At each time step the following Jacobian matrices must be evaluated.

$$\mathbf{F}_k \equiv \left. \frac{\partial \vec{f}}{\partial \vec{x}} \right|_{(\vec{x}=\hat{\vec{x}}_{k-1|k-1}, \, \vec{u}=\vec{u}_{k-1})} \tag{4.37}$$

$$\mathbf{G}_k \equiv \left. \frac{\partial \vec{f}}{\partial \vec{w}} \right|_{(\vec{x}=\hat{\vec{x}}_{k-1|k-1}, \, \vec{u}=\vec{u}_{k-1})} \tag{4.38}$$

$$\mathbf{H}_k \equiv \left. \frac{\partial \vec{h}}{\partial \vec{x}} \right|_{(\vec{x}=\hat{\vec{x}}_{k|k-1})} \tag{4.39}$$

These Jacobian matrices replace their counterparts in equations (4.14) through (4.29) with two exceptions. The mean prediction based upon the dynamic model is often accomplished with a higher order integration technique, such as a Runga Kutta algorithm, which utilizes the nonlinear dynamic model given by (3.1). Also, the $\mathbf{H}\hat{\hat{x}}_{k|k-1}$ term in (4.22), (4.23), and (4.24) is replaced with the nonlinear function given by (3.2). A compact algorithm for the EKF is located in Appendix C.

The EKF method of estimating the covariance is only an approximation. If the mean and covariance do not accurately capture the posterior statistics, the EKF solution is not optimal. However, when implemented at a high frequency with slightly nonlinear equations, this filter provides very good results.

4.3 Unscented Kalman Filter

While the EKF is effective for many applications, several limitations hinder its performance. As mentioned, the nonlinear function should exhibit nearly linear characteristics about the current operating point. If this approximation is not accurate, the state estimates may degrade or even diverge [15]. The covariance approximation technique also requires calculation of the Jacobian matrices. Often these derivatives do not exist about discontinuous operating points or their resulting values are ill conditioned. Even if the Jacobian matrix exist for highly nonlinear functions, its calculation can be very tedious and error prone, and the resulting matrix elements are often very complex and increase computational load.

The drawbacks of the EKF approximations have led to the development of a higher fidelity and derivative free mean and covariance estimation technique for extremely nonlinear functions. Central to the unscented Kalman filter, the unscented transformation addresses these deficiencies by using a sampling technique that improves covariance and mean estimates.

4.3.1 Unscented Transformation

The primary concept of the UT involves selecting a set of deterministically chosen weighted "sigma points" (vectors), which have a known mean, $\bar{\bar{x}}$, and covariance, $\mathbf{P}^{xx}$. These points are propagated through the nonlinear model, resulting in transformed points. The statistical data represented by the transformed points approximates the true PDF. Ref. [4] has shown that the

UT accurately predicts the mean and covariance to the second order, and has similar computational cost as the EKF.

Various sigma point sets exist. The set used for the UT in this research is known as the basic symmetric set. The symmetric sigma points, $\boldsymbol{\chi}$, and their associated weights are generated as follows where N_x is the dimension of the random vector.

$$\forall\; p = 1, \ldots, N_x \tag{4.40}$$

$$\boldsymbol{\chi}_{[p]} = \bar{\bar{x}} + \left(\sqrt{N_x \mathbf{P}^{xx}}\right)_{[p]} \tag{4.41}$$

$$\boldsymbol{\chi}_{[p+N_x]} = \bar{\bar{x}} - \left(\sqrt{N_x \mathbf{P}^{xx}}\right)_{[p]} \tag{4.42}$$

$$\vec{W}_{[p]} = \vec{W}_{[p+N_x]} = \frac{1}{2N_x} \tag{4.43}$$

The bracketed subscript in equations (4.41) through (4.43) represents the row or column of a matrix or the element of a vector which corresponds to the given sigma point. The term $\left(\sqrt{N_x \mathbf{P}^{xx}}\right)_{[p]}$ denotes the "pth" row or column of the matrix square root. Generally, this is calculated with a numerically stable algorithm such as the Cholesky decomposition. If the decomposition returns the matrix square root, $\mathbf{C}$, in the form $\mathbf{P} = \mathbf{C}\mathbf{C}^T$, then the columns of $\mathbf{C}$ are used. If it returns the matrix square root, $\mathbf{C}$, in the form $\mathbf{P} = \mathbf{C}^T\mathbf{C}$, then the rows are used. These sigma points have the desired mean, which is calculated by

$$\bar{\bar{x}} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \boldsymbol{\chi}_{[p]}\,, \tag{4.44}$$

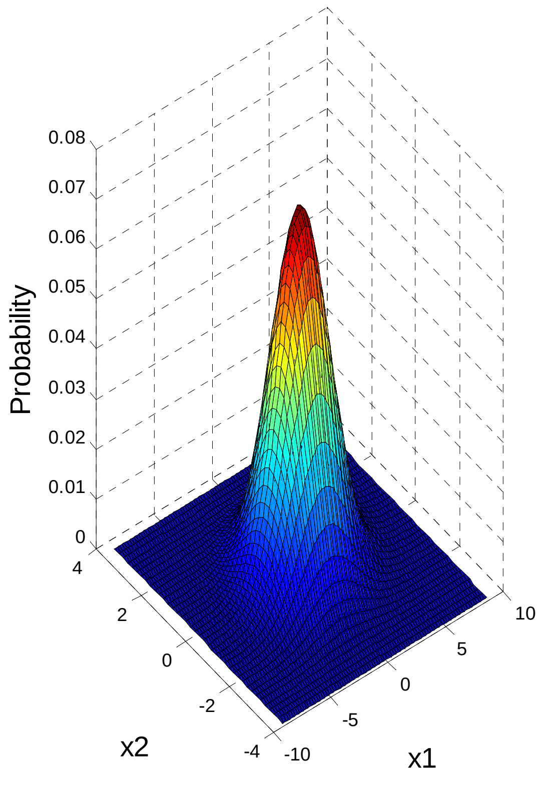
and the desired covariance, which is calculated by

$$\mathbf{P}^{xx} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \left(\boldsymbol{\chi}_{[p]} - \bar{\bar{x}}\right)\left(\boldsymbol{\chi}_{[p]} - \bar{\bar{x}}\right)^T . \tag{4.45}$$

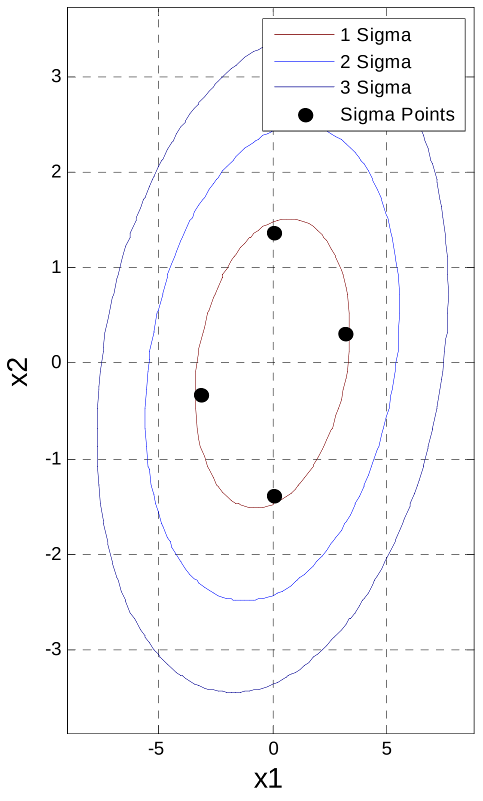
Intuitively, symmetric sigma points represent vectors in N_x dimensional space. These vectors are perturbed about the mean estimate by an amount that properly models the mean and covariance. These concepts are illustrated in Fig. 4-2 for a two element random vector with mean values and covariance matrix given in Table 4-1. In (a), the PDF for two random variables x_1 and x_2 are plotted with the probability along the z axis. Three contour lines and sigma points are plotted in (b). The innermost contour line denotes the value in which 68.2% of the variables should lie assuming a Gaussian distribution. The middle and outer contour lines correspond to 95.5% and 99.7% confidence intervals, respectively. The sigma points for this plot are generated using (4.40) through (4.43). The actual points represent the end of vectors, which originate from the mean values.

Table 4-1: Mean and Covariance Matrix for Random Variables

Variables	Mean	Covariance Matrix
$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\bar{\vec{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\mathbf{P}^{xx} = \begin{bmatrix} 5 & .5 \\ .5 & 1 \end{bmatrix}$



(a)



(b)

Figure 4-2: (a) The PDF of Two Random Variables x1 and x2, and (b) the Corresponding Probability Ellipses and Sigma Points

After the sigma points are generated, they are propagated through a nonlinear function, $\vec{h}(\cdot)$, which yields the transformed points, ζ.

$$\zeta_{[p]} = \sum_{p=1}^{2N_x} \vec{h}\left(\chi_{[p]}\right) \tag{4.46}$$

The mean is estimated with

$$\bar{\bar{z}} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \zeta_{[p]} , \tag{4.47}$$

and transformed covariance is found by

$$\mathbf{P}^{zz} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \left(\zeta_{[p]} - \bar{\bar{z}}\right)\left(\zeta_{[p]} - \bar{\bar{z}}\right)^T . \tag{4.48}$$

While this mean and covariance estimation technique is conceptually simple, its effectiveness for capturing the posterior statistics of nonlinear functions is superior to the first order approximation used in the EKF in many situations. Also, the simple symmetric set presented in detail here only represents one set of sigma points. If higher order moment information is known about the error distribution of a RV, other advanced sigma points sets exist, which capture these higher moments. For a thorough description of other advanced sigma points see [4].

4.3.1.1 Mean and Covariance Estimation Example

A simple nonlinear mean and covariance estimation example is presented here to demonstrate the accuracy gains of the UT over the traditional linear approximations utilized in the EKF. The example, outlined in Fig. 4-3, consists of the conversion from polar to Cartesian coordinates. The current mean values for two Gaussian random variables Θ and r and their associated covariance matrix are assumed known. Mean and covariance estimates for x and y are desired. This problem is often encountered in state estimation as many sensors, such as radar and laser rangefinders, measure a range with respect to a relative bearing. The nonlinearities become increasingly apparent as the variance on the angle increases, and thus, this problem is often studied in nonlinear estimation literature [4].

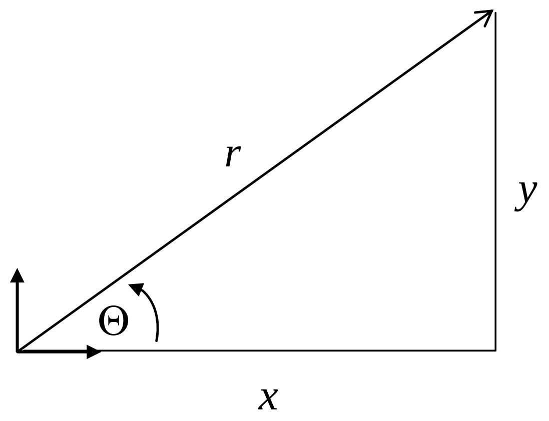


Figure 4-3: Simple Nonlinear Example

Table 4-2 describes the initial mean vector values and initial covariance matrix used for this demonstration where

$$\vec{x} = \begin{bmatrix} \Theta \\ r \end{bmatrix}. \tag{4.49}$$

Note that Θ is given in radians while the units for r, x, and y are the same but arbitrary.

Table 4-2: Initial Mean Vector and Covariance Matrix

Mean	**Covariance Matrix**
$\bar{\vec{x}} = \begin{bmatrix} .7854 \\ 5 \end{bmatrix}$	$\mathbf{P}^{xx} = \begin{bmatrix} .1 & 0 \\ 0 & .05 \end{bmatrix}$

The desired output vector consists of x and y

$$\vec{z} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.50}$$

and is calculated with the following nonlinear output model.

$$\vec{h}(\Theta, r) = \begin{bmatrix} r\cos(\Theta) \\ r\sin(\Theta) \end{bmatrix} \tag{4.51}$$

Figure 4-4 displays the results of the EKF linearization, UT symmetric sigma points, and Monte Carlo mean and covariance approximations. The Monte Carlo method provides a benchmark comparison for the other two estimation techniques. As in Fig. 4-2 (b), the ellipses in the first row of plots represent the true 68.2%, 95.5%, and 99.7% confidence intervals because Θ and r are Gaussian. The sigma points are denoted with solid black circles as shown in the middle column plots, (b), while the plus signs in the rightmost column of plots, (c), denote Monte Carlo points. The circle near the middle of the ellipse contours marks the mean values. The first row of plots represents the initial mean and covariance of the random variables as given in Table 4-2. Both the symmetric sigma points and Monte Carlo generated points are displayed on their corresponding graphs to emphasize how they model the initial Gaussian statistics. The last row of plots represents the mean and covariance of the random variables after undergoing the nonlinear transformation where the symmetric sigma points and Monte Carlo points are plotted again to provide visual understanding. The true distributions are no longer Gaussian after transformation, and the confidence interval ellipses in those corresponding figures are used only to represent crude approximations of probability.

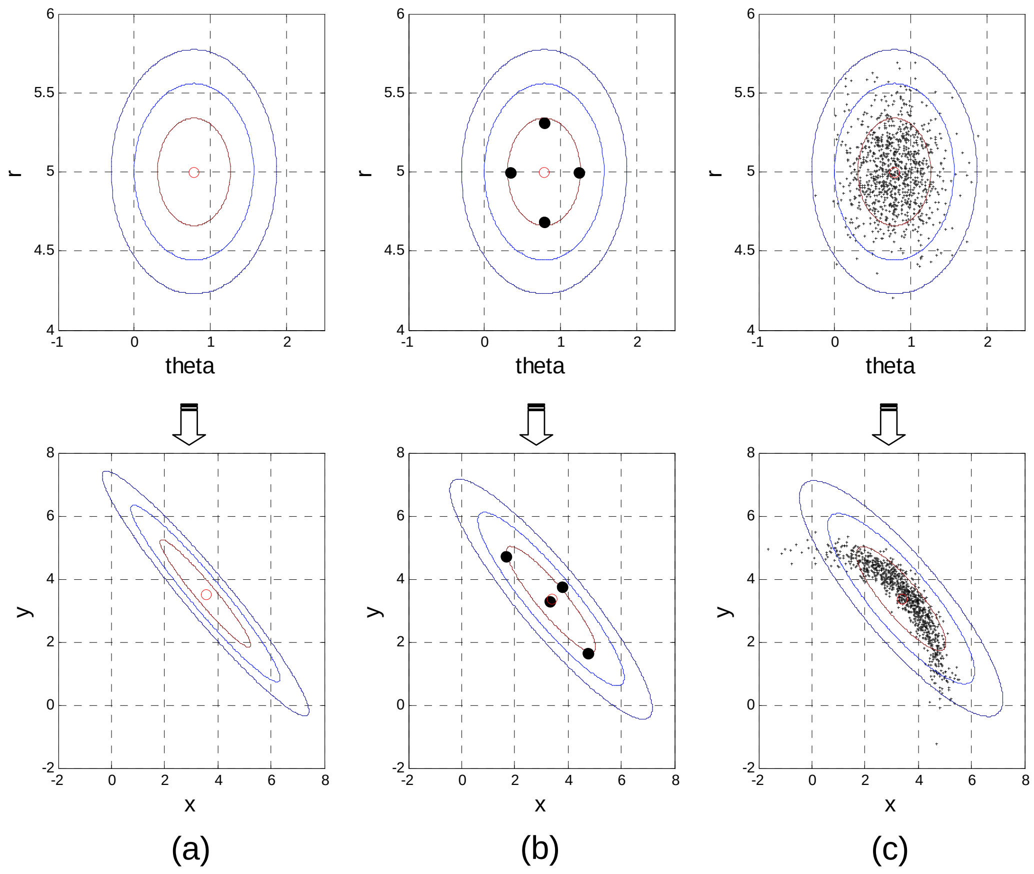


Figure 4-4: Mean and Covariance Transformation Using (a) EKF Linearization Approximation, (b) UT Symmetric Sigma Points, and (c) Monte Carlo Sampling

The actual numerical values for the means and covariance matrices are located in Table 4-3. The EKF linearization uses (4.33) and (4.35) to calculate these values where

$$\left.\frac{\partial\left(\bar{h}(\bar{x})\right)}{\partial\bar{x}}\right|_{\bar{x}=\bar{\bar{x}}} = \begin{bmatrix} -r\sin(\Theta) & \cos(\Theta) \\ r\cos(\Theta) & \sin(\Theta) \end{bmatrix}. \tag{4.52}$$

The UT uses (4.40) through (4.43) to generate a symmetric set of sigma points where $N_x = 2$. The points are transformed via (4.51), upon which the mean and covariance are determined by (4.47) and (4.48). The Monte Carlo (MC) points are generated with MATLAB's $randn(\cdot)$ function. These points are randomly generated with a zero mean and unit normal distribution. In order to get the desired covariance and mean, the generated points are scaled by the matrix square root, i.e. the Cholesky decomposition, of $\mathbf{P}^{xx}$ and then shifted by the mean value.

$$MC\ points = \mathbf{C} * randn(vector\ size, num\_sample) + \bar{\bar{x}} \tag{4.53}$$

where $\mathbf{C}$ is the matrix square root of $\mathbf{P}^{xx}$ in the form

$$\mathbf{P}^{xx} = \mathbf{C}\mathbf{C}^T \tag{4.54}$$

After transformation through the output model, the posterior statistics are determined using MATLAB's $mean(\cdot)$ and $\operatorname{cov}(\cdot)$ functions.

Table 4-3: Transformed Mean Vector and Covariance Matrix

Estimation Technique	**Mean**	**Covariance**
EKF Linearization Approximation	$\bar{\bar{z}} = \begin{bmatrix} 3.5355 \\ 3.5355 \end{bmatrix}$	$\mathbf{P}^{zz} = \begin{bmatrix} 1.2750 & -1.2250 \\ -1.2250 & 1.2750 \end{bmatrix}$
UT Symmetric Sigma Points	$\bar{\bar{z}} = \begin{bmatrix} 3.3617 \\ 3.3617 \end{bmatrix}$	$\mathbf{P}^{zz} = \begin{bmatrix} 1.2241 & -1.1136 \\ -1.1136 & 1.2241 \end{bmatrix}$
Monte Carlo 1000 Points	$\bar{\bar{z}} = \begin{bmatrix} 3.3589 \\ 3.3773 \end{bmatrix}$	$\mathbf{P}^{zz} = \begin{bmatrix} 1.2184 & -1.0242 \\ -1.0242 & 1.1767 \end{bmatrix}$

For this particular example, the UT clearly has a distinct advantage over the EKF linearization technique for both mean and covariance estimation. While both the UT and the EKF linearization overestimate covariance matrix values, the UT generates results closer to the sampled estimate. The same is true for the mean values. Even though the UT provides a better estimate, it is still an approximation technique as is evident during comparison with Monte Carlo results. However, the tradeoff between accuracy and computational load makes this method appealing for practical applications such as real-time state estimation.

4.3.2 Unscented Kalman Filter Algorithm

Julier and Uhlmann modified the basic Kalman Filter algorithm to incorporate the UT's ability to improve mean and covariance estimates for nonlinear systems [4]. A detailed version of this algorithm, which uses symmetric sigma points, is shown in this section for state equations given in the form of (3.1) and (3.2). This basic filter is also used as the core estimator for this research due to the nonlinear nature of the state equations presented in Chapter 3, especially the feature observation model.

```
**FOR** $k = 1, \ldots, \infty$
    Form augmented state vector and covariance matrix:
    $\hat{\vec{x}}^{a}_{k-1|k-1} = \left[ \left( \hat{\vec{x}}^{x}_{k-1|k-1} \right)^{T} \quad \vec{w}^{T}_{k-1} \right]^{T}$    (4.55)
    $\mathbf{P}^{a}_{k-1|k-1} = E\left\{ \left( \hat{\vec{x}}^{a}_{k-1|k-1} - \vec{x}^{a}_{k-1} \right) \left( \hat{\vec{x}}^{a}_{k-1|k-1} - \vec{x}^{a}_{k-1} \right)^{T} \right\}$    (4.56)
    $\mathbf{P}^{a}_{k-1|k-1} = \begin{bmatrix} \mathbf{P}^{xx}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{bmatrix}$    (4.57)
    $N_x = L_{\hat{\vec{x}}^{x}} + L_{\vec{w}}$    (4.58)
    Generate sigma points:
    **FOR** $p = 1, \ldots, N_x$
        $\boldsymbol{\chi}^{a}_{[p]} = \hat{\vec{x}}^{a}_{k-1|k-1} + \left( \sqrt{N_x \mathbf{P}^{a}_{k-1|k-1}} \right)_{[p]}$    (4.59)
        $\boldsymbol{\chi}^{a}_{[p+N_x]} = \hat{\vec{x}}^{a}_{k-1|k-1} - \left( \sqrt{N_x \mathbf{P}^{a}_{k-1|k-1}} \right)_{[p]}$    (4.60)
        $\vec{W}_{[p]} = \vec{W}_{[p+N_x]} = \frac{1}{2N_x}$    (4.61)
        where $\boldsymbol{\chi}^{a}_{[p]} = \left[ \left( \boldsymbol{\chi}^{x}_{[p]} \right)^{T} \quad \left( \boldsymbol{\chi}^{w}_{[p]} \right)^{T} \right]^{T}$    (4.62)
    **END FOR**
    Propagate sigma points through nonlinear dynamic model:
    **FOR** $p = 1, \ldots, 2N_x$
        $\boldsymbol{\chi}^{x}_{[p]} = \int_{k-1}^{k} \vec{f}\left( \boldsymbol{\chi}^{x}_{[p]}, \vec{u}_{k-1}, \boldsymbol{\chi}^{w}_{[p]} \right) dt$    (4.63)
    **END FOR**
    Determine mean and covariance of sigma points:
    $\hat{\vec{x}}^{x}_{k|k-1} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \boldsymbol{\chi}^{x}_{[p]}$    (4.64)
    $\mathbf{P}^{xx}_{k|k-1} = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \left( \boldsymbol{\chi}^{x}_{[p]} - \hat{\vec{x}}^{x}_{k|k-1} \right) \left( \boldsymbol{\chi}^{x}_{[p]} - \hat{\vec{x}}^{x}_{k|k-1} \right)^{T}$    (4.65)
    **IF (**Output Measurement Update**)**
        Form augmented state vector and covariance matrix:
        $\hat{\vec{x}}^{a}_{k|k-1} = \left[ \left( \hat{\vec{x}}^{x}_{k|k-1} \right)^{T} \quad \vec{v}^{T}_{k} \right]^{T}$    (4.66)
        $\mathbf{P}^{a}_{k|k-1} = E\left\{ \left( \hat{\vec{x}}^{a}_{k|k-1} - \vec{x}^{a}_{k} \right) \left( \hat{\vec{x}}^{a}_{k|k-1} - \vec{x}^{a}_{k} \right)^{T} \right\}$    (4.67)
        $\mathbf{P}^{a}_{k|k-1} = \begin{bmatrix} \mathbf{P}^{xx}_{k|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{k} \end{bmatrix}$    (4.68)
        $N_x = L_{\hat{\vec{x}}^{x}} + L_{\vec{v}}$    (4.69)
        Generate sigma points:
        **FOR** $p = 1, \ldots, N_x$
            $\boldsymbol{\chi}^{a}_{[p]} = \hat{\vec{x}}^{a}_{k|k-1} + \left( \sqrt{N_x \mathbf{P}^{a}_{k|k-1}} \right)_{[p]}$    (4.70)
            $\boldsymbol{\chi}^a_{[p+N_x]} = \hat{\vec{x}}^a_{k|k-1} - \left(\sqrt{N_x \mathbf{P}^a_{k|k-1}}\right)_{[p]}$    (4.71)
            $\vec{W}_{[p]} = \vec{W}_{[p+N_x]} = \frac{1}{2N_x}$    (4.72)
            where $\boldsymbol{\chi}^a = \left[(\boldsymbol{\chi}^x)^T \quad (\boldsymbol{\chi}^v)^T\right]^T$    (4.73)
        END FOR
        Propagate sigma points through nonlinear output model:
        FOR $p = 1, \ldots, 2N_x$
            $\boldsymbol{\zeta}_{[p]} = \vec{h}\left(\boldsymbol{\chi}^x_{[p]}, \boldsymbol{\chi}^v_{[p]}\right)$    (4.74)
        END FOR
        Determine mean and covariance of sigma points:
        $\hat{\vec{z}}_k = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \boldsymbol{\zeta}_{[p]}$    (4.75)
        $\mathbf{P}^{zz}_k = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \left(\boldsymbol{\zeta}_{[p]} - \hat{\vec{z}}_k\right)\left(\boldsymbol{\zeta}_{[p]} - \hat{\vec{z}}_k\right)^T$    (4.76)
        $\mathbf{P}^{xz}_k = \sum_{p=1}^{2N_x} \vec{W}_{[p]} \left(\chi^x_{[p]} - \hat{\vec{x}}^x_{k|k-1}\right)\left(\boldsymbol{\zeta}_{[p]} - \hat{\vec{z}}_k\right)^T$    (4.77)
        Correct estimates:
        $\mathbf{K}_k = \mathbf{P}^{xz}_k \left(\mathbf{P}^{zz}_k\right)^{-1}$    (4.78)
        $\hat{\vec{x}}^x_{k|k} = \hat{\vec{x}}^x_{k|k-1} + \mathbf{K}_k \left(\tilde{\vec{z}}_k - \hat{\vec{z}}_k\right)$    (4.79)
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1} - \mathbf{K}_k \mathbf{P}^{zz}_k \mathbf{K}^T_k$    (4.80)
    ELSE
        $\hat{\vec{x}}^x_{k|k} = \hat{\vec{x}}^x_{k|k-1}$    (4.81)
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1}$    (4.82)
    END IF
END FOR
```

This algorithm needs additional explanation. One of the most notable distinctions between the UKF algorithm and traditional KF and EKF algorithms is the creation of an augmented state vector, (4.55) and (4.66), and covariance matrix, (4.56) and (4.67), during both the start of the prediction and correction phases. The mean vector augmentation must include the current state vector estimates and input, $\vec{w}_{k-1}$, or output, $\vec{v}_k$, noise vectors. The mean of the noise vectors do not have to be zero. They can be modified to reflect known biases (i.g. thermally induced biases). This augmentation is required since both the nonlinear dynamic and observation models are functions of random noise vectors. In order to account for the random noise influence upon the uncertainty of the state estimates, sigma points must be created for these random variables, which is also the reason an augmented covariance matrix $\mathbf{P}^a$ is formed. Equation (4.58) represents the determination of the length of the augmented vector where $L_{\hat{\vec{x}}^x}$ is the length of the state estimate vector and $L_{\vec{w}}$ is the length of the input noise vector. The next major step involves calculation of the simple symmetric sigma points. The points are then propagated through the nonlinear dynamic model. For this research, a fourth order Runga Kutta algorithm is used for integration due to its accuracy over standard Euler integration. The new mean and state error covariance matrix are then approximated with the aid of (4.64) and (4.65). This process continues until a measurement update is available. When a measurement is available, a new set of sigma points are formed with the new augmented vector and covariance matrix. This new vector and matrix may be a different size than those used during the prediction steps, which is why N_x must be recalculated in (4.69) where $L_{\vec{v}}$ is the length of the output noise vector. Following the sigma point propagation through the nonlinear output model, the new output mean $\hat{\vec{z}}_k$, and covariance matrix, $\mathbf{P}_k^{zz}$, are determined. After calculation of the cross covariance, $\mathbf{P}_k^{xz}$, the Kalman gain is found. Finally, the state estimate and error covariance estimate are corrected based upon the Kalman gain. Equations (4.78) through (4.80) are actually the same correction equations as used in both the KF and EKF where the cross covariance matrix, $\mathbf{P}_{k|k-1}^{xx}\mathbf{H}^T$, is replaced by $\mathbf{P}_k^{xz}$ and the innovation covariance, $\mathbf{H}\mathbf{P}_{k|k-1}^{xx}\mathbf{H}^T + \mathbf{R}_k$, is given by $\mathbf{P}_k^{zz}$. However, the covariance estimates are found with the UT instead of linearization approximation techniques.

Chapter 5: Observability Issues

This chapter discusses the observability of the state equations developed for the aircraft, feature, and target. A set of state equations is said to be observable when knowledge of the input and output over a finite time interval allows unique determination of the initial state vector [9]. A traditional test for linear systems analyzes the rank of the observability matrix. If the rank of the observability matrix is greater to or equal to the number of states, the system is considered observable. Observability is the fundamental requirement for any observer, as the loss of observability corresponds to the inability to generate converging state estimates.

Nonlinear equations, unfortunately, have to be linearized before the observability matrix can be formed. The analytical observability matrix developed from the first order term of the Taylor series expansion of the nonlinear equations will contain elements dependent upon current states and inputs. Therefore, in certain instances, observability may be lost for some states depending upon the current operating point. This is not a problem so long as the state does not persist in this condition. For example, the GPS/INS is unobservable when the accelerations are zero. The state frequently passes through/near this condition, but the estimator still works very well. However, a major problem occurs when the unobservable condition is not a function of a particular state but rather a fundamental problem of the system itself. In this case determining rank of the observability matrix from the linearized system does not necessarily lead to physical intuition of observability issues. This thesis provides an intuitive observability explanation based upon heuristic arguments developed during simulations.

5.1 Feature Observability

The proposed target tracking solution requires information obtained from the 3D location of features, and therefore, their observability is analyzed here. Because the camera poses are consistent with the position and orientation of the aircraft, the state equations discussed here include both the aircraft and feature states, thus defining the SLAM problem.

$$\dot{\vec{x}} = \begin{bmatrix} \dot{\vec{x}}_a \\ \dot{\vec{x}}_f \end{bmatrix} \tag{5.1}$$

$$\vec{z} = \begin{bmatrix} \vec{z}_a \\ \vec{z}_f \end{bmatrix} \tag{5.2}$$

Fig. 5-1 illustrates the relationships between the aircraft states and feature states through multiple observations. At time $(k-1)\Delta t$, the UAV, denoted with a solid outline, flies directly over a stationary feature and captures an image.

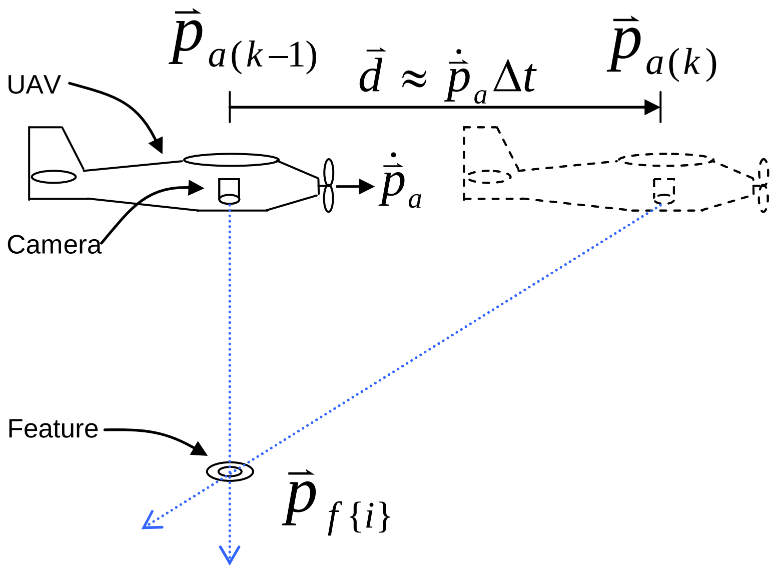


Figure 5-1: The Triangulation of a Feature as Viewed from Two Different Poses

The vector pointing from the camera through the actual feature location denotes the line in space corresponding to the family of possible, but still unknown, 3D feature locations. A short time later, Δt, the UAV, now denoted by its dashed outline, has traveled a small distance and captures another image. A new vector, pointing from the new camera location to the feature, symbolizes the line along which the feature should reside. Assuming the feature's location did not change, these two vectors fully triangulate the feature's location. However, the scale of the resulting triangle remains unknown. The scale of this triangle can be determined if the distance vector, $\vec{d}$, pointing from the previous aircraft position to the current position, is observed with GPS. Realistically, the measurement of these two vectors is corrupted by noise and uncertainty among the state estimates, resulting in skew vectors with some residual error in the feature's location. With sufficient measurements taken from different aircraft positions and orientations, the mean location of the feature in 3D space can be estimated accurately. The camera, and hence, the aircraft must move to allow triangulations and residuals corrections to exist.

5.2 Target Observability

The state equations necessary for the examination of observability for the target tracking model must contain both the aircraft and target state equations because the target measurement depends upon the aircraft states in much the same way as the feature observations did.

$$\dot{\vec{x}} = \begin{bmatrix} \dot{\vec{x}}_a \\ \dot{\vec{x}}_t \end{bmatrix} \tag{5.3}$$

$$\vec{z} = \begin{bmatrix} \vec{z}_a \\ \vec{z}_t \end{bmatrix} \tag{5.4}$$

The major difference between target estimation and feature estimation originates from the dynamic nature of the target. This is illustrated in Fig. 5-2.

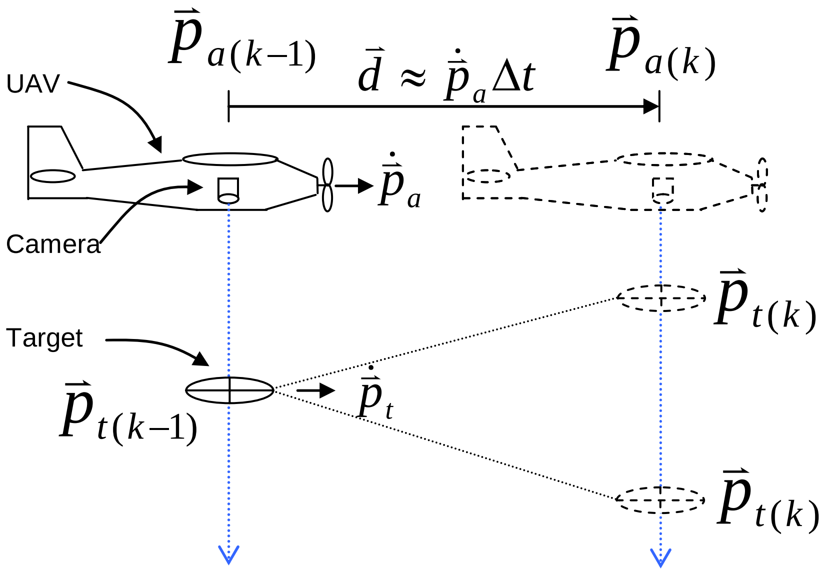


Figure 5-2: The Lack of Triangulation of a Moving Target as Viewed from Two Different Poses

Once again, the vector from the camera passing through the target represents the family of possible target locations. The target, in this case, has a velocity component near the same magnitude and direction as the aircraft, resulting in the same target images. Triangulation of the target's position is not possible. The target's velocity also retains ambiguity along the direction of observation, as illustrated by the two dashed target positions in Fig. 5-2. Simulations based upon the dynamic and observation models of (5.3) and (5.4) have supported this heuristic argument. A moving vehicle cannot observe states of another moving vehicle from monocular vision alone.

Chapter 6: Novel Solution

The ground target constraint is used to extrapolate a 3D measurement from the 2D image coordinates to overcome the unobservability problems for monocular tracking of moving targets. The general target observation model is redefined as illustrated in Fig. 6-1, where the target is assumed to lie in a plane defined by nearby features.

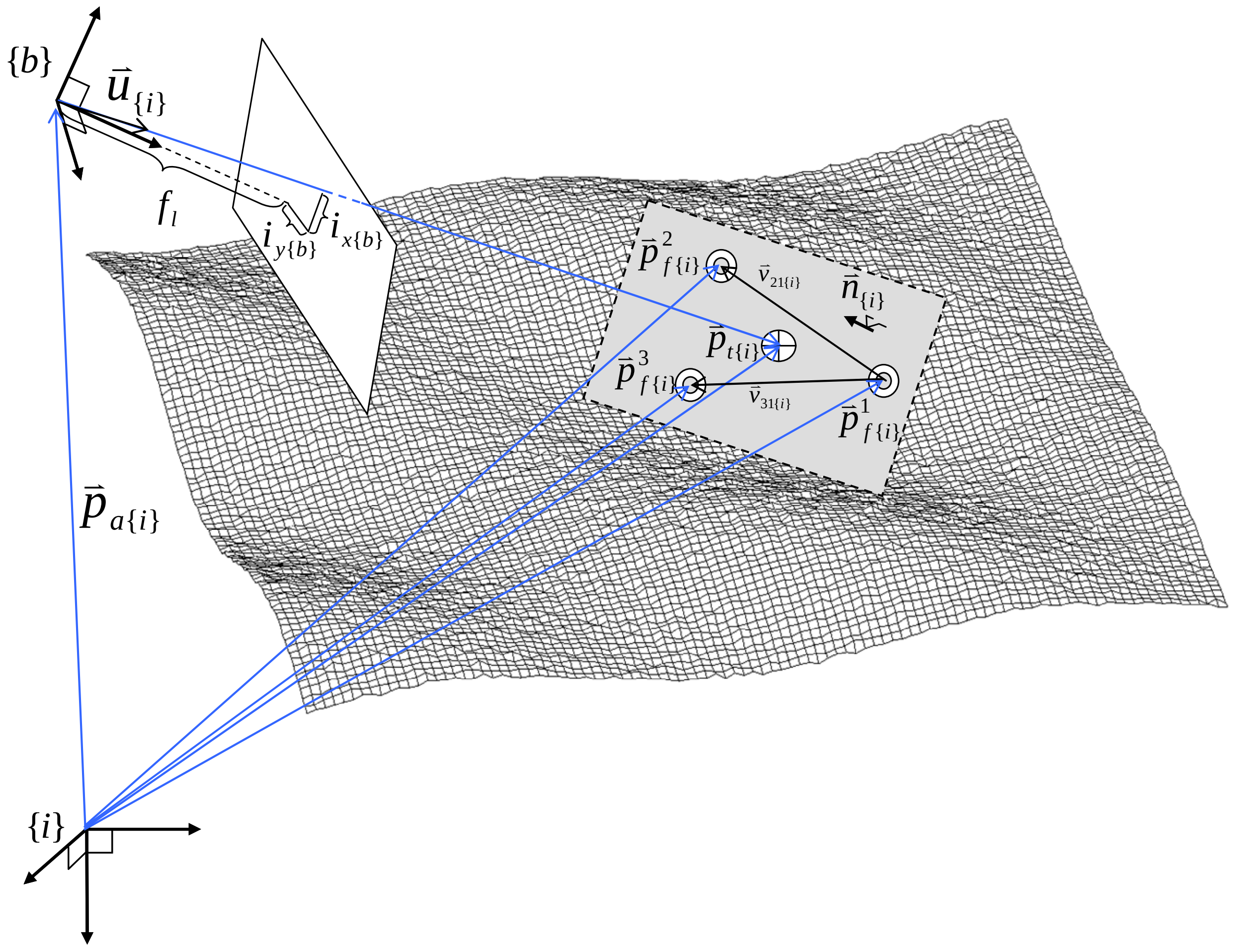


Figure 6-1: The Relationships for the Redefined Target Observation Model

In Fig. 6-1, the ground plane is defined by the three closest features to the target. The normal vector for the ground plane is given by

$$\vec{n}_{\{i\}} = \vec{v}_{21\{i\}} \times \vec{v}_{31\{i\}} \tag{6.1}$$

where

$$\vec{v}_{21\{i\}} = \vec{p}^{\,2}_{f\{i\}} - \vec{p}^{\,1}_{f\{i\}} \tag{6.2}$$

$$\vec{v}_{31\{i\}} = \vec{p}^{\,3}_{f\{i\}} - \vec{p}^{\,1}_{f\{i\}} . \tag{6.3}$$

The unit vector pointing from the camera toward the target is found using the image of the target and the aircraft state. This vector is expressed in the inertial frame as follows.

$$\vec{u}_{\{i\}} = \mathbf{R}_{\{ib\}} \frac{[i_{x\{b\}} \quad i_{y\{b\}} \quad f_l]^T}{\left\| [i_{x\{b\}} \quad i_{y\{b\}} \quad f_l] \right\|} \tag{6.4}$$

The position of the target, $\vec{p}_{t\{i\}}$, is now written as a vector sum

$$\vec{p}_{t\{i\}} = \lambda \vec{u}_{\{i\}} + \vec{p}_{a\{i\}} \tag{6.5}$$

where λ is an unknown scalar at this point. To solve for this scalar quantity, another equation is formed based on the fact the dot product of any vector in the plane with the normal vector, $\vec{n}^{T}_{\{i\}}$, must be zero. This includes the vector from any feature to the target.

$$\vec{n}^{T}_{\{i\}} (\vec{p}_{t\{i\}} - \vec{p}^{\,1}_{f\{i\}}) = 0 \tag{6.6}$$

Substituting (6.5) into (6.6) and rearranging results in the following.

$$\lambda \vec{n}^{T}_{\{i\}} \vec{u}_{\{i\}} = \vec{n}^{T}_{\{i\}} (\vec{p}^{\,1}_{f\{i\}} - \vec{p}_{a\{i\}}) \tag{6.7}$$

Solving (6.7) for λ gives

$$\lambda = \frac{\vec{n}^{T}_{\{i\}} (\vec{p}^{\,1}_{f\{i\}} - \vec{p}_{a\{i\}})}{\vec{n}^{T}_{\{i\}} \vec{u}_{\{i\}}} . \tag{6.8}$$

Substituting (6.8) into (6.5) results in an expression for the target location.

$$\vec{p}_{t\{i\}} = \left[\frac{\vec{n}^{T}_{\{i\}} (\vec{p}^{\,1}_{f\{i\}} - \vec{p}_{a\{i\}})}{\vec{n}^{T}_{\{i\}} \vec{u}_{\{i\}}} \right] \vec{u}_{\{i\}} + \vec{p}_{a\{i\}} \tag{6.9}$$

All of the terms on the right side of (6.9) are obtained from the camera sensor and estimated SLAM states. By providing a position estimate in 3D space, the observability issue for the ground target is eliminated. The measurement equation for the target, in (3.32), is thus modified to consist of a target position measurement with noise.

$$\vec{z}_t = \vec{p}_{t\{b\}} = \vec{p}_{tm\{i\}} - \vec{v}_{p_t\{i\}} \tag{6.10}$$

The covariance of the noise term, $\vec{v}_{p_t\{i\}}$, in the target position measurement can be estimated using the UT and the covariance of the feature estimates, aircraft estimate, and camera coordinates used in (6.1), (6.4), and (6.9).

6.1 Loosely Coupled Estimation Solution

While the UT provides a method for determining a mean measurement with predicted covariance, care must be taken when applying the target observation model given by (6.10) within a tightly coupled KF, EKF, or UKF. Tightly coupled, in this section, refers to the combination of target states and SLAM states within a single filter. The modified target position measurement is a function of other estimated states and its covariance is dependent upon current state covariance estimates. A fundamental assumption made within the derivation of the KF is that the measurement noise is white and orthogoanal to the estimated state covariance [10]. By using the target observation model defined by (6.10) instead of (3.32), correlation is induced. This issue is avoided by estimating the target states with another loosely coupled KF, although this does not guarantee that the output target measurement is a white signal. However, any autocorrelation of the target measurement signal does not adversely influence the aircraft and feature state estimates in this loosely coupled setup. Fig. 6-2 illustrates the high-level perspective of the loosely coupled estimators and their relationships.

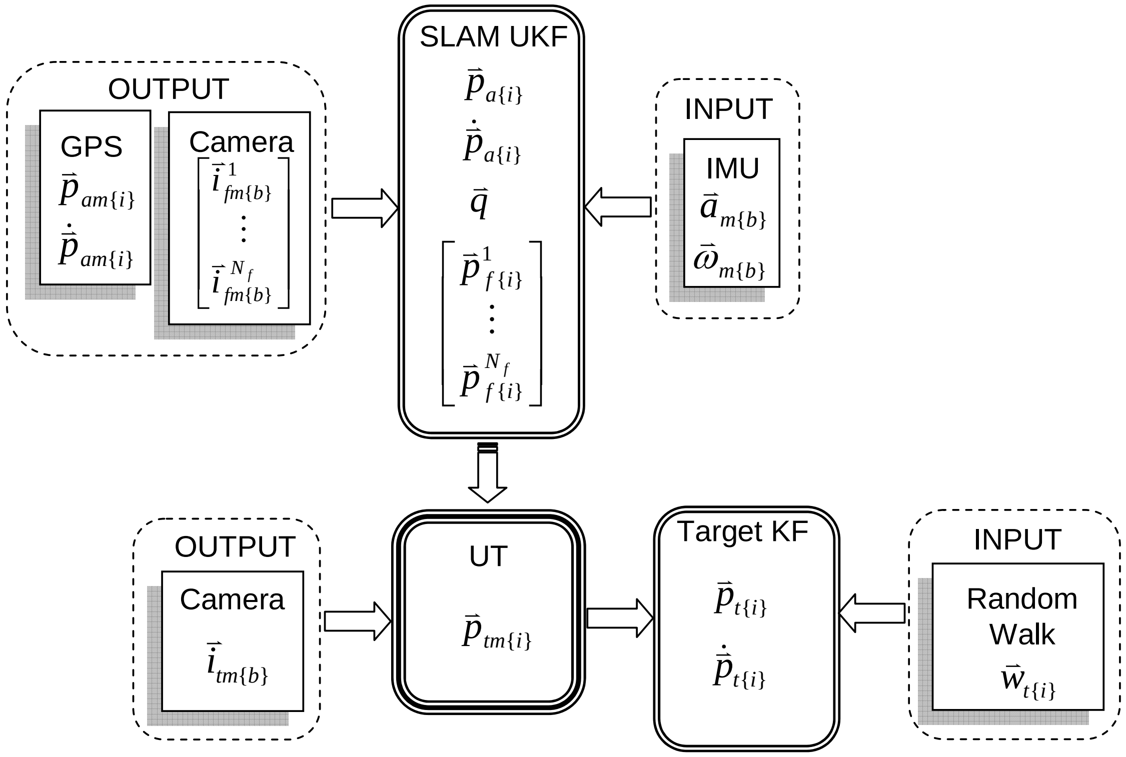


Figure 6-2: The Relationships between the SLAM UKF, UT, and the Target KF

The SLAM model utilizes the state equations given in (5.1) and (5.2). A UKF is employed to estimate the aircraft and feature states due to the noise terms and nonlinear nature of the state equations. The input and output measurements are contained in dashed boxes, which represent the conversion from continuous to discrete time. The output of this SLAM UKF estimator, including the estimated values for aircraft position, aircraft orientation, and feature locations, becomes one input to the UT block. Another input to this block includes the output measurements of the target location in image space coordinates. The UT then converts the input means

$$\vec{x}_{ut} = \begin{bmatrix} \vec{p}_{a\{i\}} \\ \vec{q} \\ \vec{p}^{\,1}_{f\{i\}} \\ \vec{p}^{\,2}_{f\{i\}} \\ \vec{p}^{\,3}_{f\{i\}} \\ \vec{i}_{tm\{b\}} \end{bmatrix} \tag{6.11}$$

and covariance matrix

$$\mathbf{P}^{x_{ut}x_{ut}} = E\left\{\left(\vec{x}_{ut} - \bar{\vec{x}}_{ut}\right)\left(\vec{x}_{ut} - \bar{\vec{x}}_{ut}\right)^T\right\} \tag{6.12}$$

into a target position measurement and covariance matrix estimate via the nonlinear relationships defined by (6.1), (6.4), and (6.9). The output of the UT block becomes the output measurement for the target KF. The input to the target model is simply the random walk zero mean noise term. A basic Kalman Filter is used for the target estimator, because the target state equations are linear.

6.2 Mean and Covariance Extraction and Inclusion

Extracting desired mean values from an estimator is trivial. For the UT block in the previous section, a simple orthogonal linear transformation, $\mathbf{T}$, is defined which relates the SLAM UKF state estimate vector to the first five vectors in $\vec{x}_{ut}$. This matrix, which consists solely of ones and zeros, must continually be updated since the three features that describe the ground plane about the target change. The required position and quaternion vectors in $\vec{x}_{ut}$ are then found by pre-multiplication of the SLAM state vector by $\mathbf{T}$. The associated covariance terms must also be extracted from the SLAM UKF state covariance matrix, $\mathbf{P}^{xx}$. Since the mean transformation matrix is linear, linear covariance propagation techniques apply. The upper diagonal block of $\mathbf{P}^{x_{ut}x_{ut}}$ is then found by the following transformation

$$\mathbf{T}\mathbf{P}^{xx}\mathbf{T}^T . \tag{6.13}$$

Finally, $\vec{x}_{ut}$ is completed with concatenation of the target pixel measurements. The pixel noise covariance matrix is also augmented to the covariance matrix determined from (6.13), giving $\mathbf{P}^{x_{ut}x_{ut}}$. This covariance matrix is block diagonal, which is required to assure that it is positive semidefinite.

The basic process of extracting and augmenting mean and covariance information from the filter is a common necessity for a variety of applications, including feature initialization and loosely coupled estimator interaction. Any future reference regarding the manipulation of mean and covariance information follows the guidelines listed in this section and are not explained in detail.

Chapter 7: Simulation

Simulation has been used to test the validity of the SLAM and target tracking estimators discussed previously. The analysis includes assessment of the convergence of feature locations, the aircraft state estimation accuracy gained by SLAM estimation techniques over the typical GPS/INS estimation, and the ability to track ground targets.

7.1 Environment and Setup

The simulation requires modeling of an aircraft's dynamic equations, and the generation of IMU, GPS, and camera data for filter estimation. Since aerodynamic coefficients and inertial values for the ECat UAV are unknown, a common six DOF Navion aircraft model, taken from [11], is implemented in a MATLAB/SIMULINK S-Function. The steady level flight conditions were found using MATLAB's trim function. At a 100m altitude, the equilibrium trim values were a 44.7m/s velocity, a 2.36° pitch angle, a -1.73° elevator deflection, and 1,331N of propeller thrust. A simple autopilot, consisting of lateral and longitudinal dynamic controllers designed about the trim conditions, allows tracking of commands from a straight-line waypoint navigator. The S-Function, autopilot controllers, and navigator are all housed in the leftmost block of Fig. 7-1. The vector outputs from this block include the aircraft velocity (m/s), angular rates (rad/s), Euler angles (rad), position (m), and body axes accelerations (m/s^2). Three of these output vectors are routed to and stored in the "Actual" states variable for plotting analysis and filter performance comparisons.

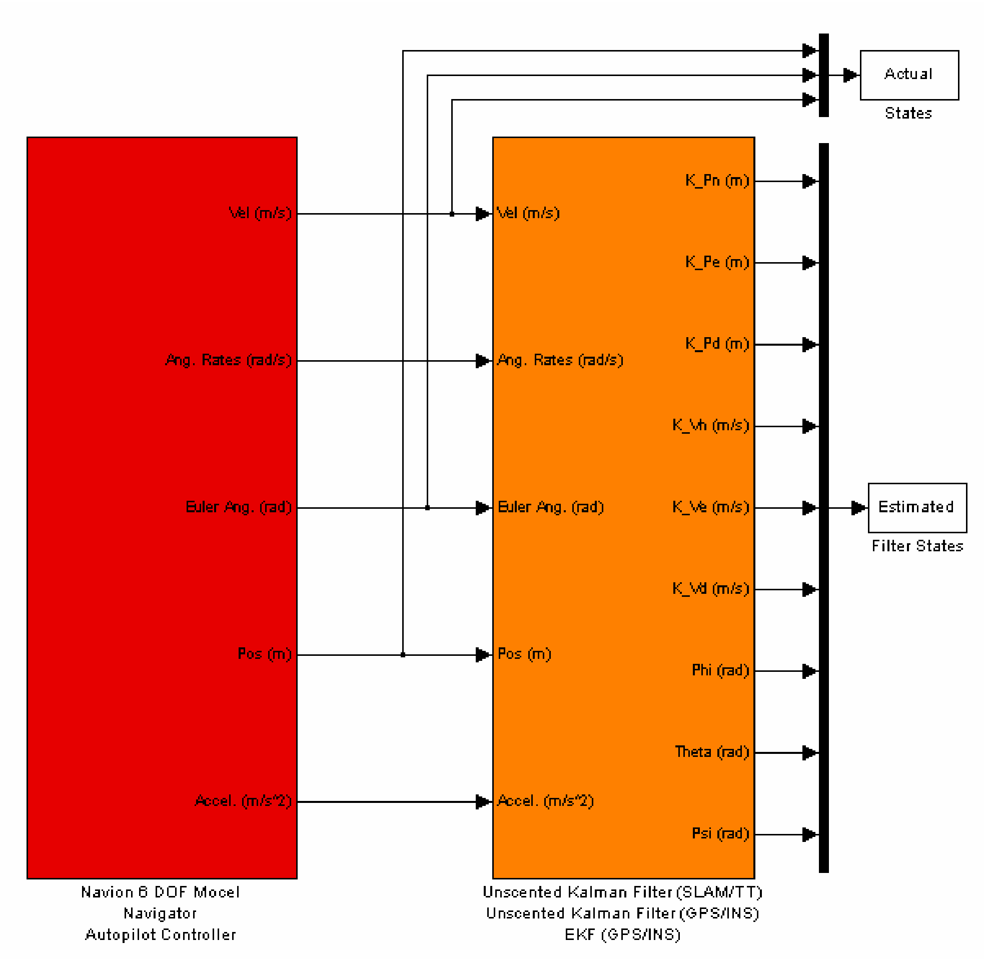


Figure 7-1: Target Tracking Estimator Simulink Diagram

All aircraft outputs become inputs to the filter block. The filter subsystem includes three different discrete time S-Functions. The main S-Function includes the proposed loosely coupled UKF SLAM and Target KF estimators. The next two S-Functions, one utilizing a UKF and the other a EKF, both generate a typical aircraft GPS aided INS for benchmark comparisons. All three filters require 25Hz IMU input data and 5Hz GPS correction data. In order to generate accelerometer and gyro IMU measurements, the aircraft's true accelerations and angular rates are corrupted with zero mean Gaussian white noise. The aircraft's position and velocity outputs are also corrupted to construct GPS measurements. Several of the noise statistics, displayed in Table 7-1, were approximated from a commercial grade UAV IMU and GPS chipset. All noise vector elements are assumed independent and therefore their covariance matrices are diagonal. Independence between the noise vectors themselves is also assumed. The UKF SLAM filter also requires uncorrupted position and Euler angle inputs to produce camera images. The estimated position, velocity, and orientation are outputted and stored in the "Estimated" variable.

Table 7-1: Noise Statistics

Location	**Noise Source**	**Notation**	**Covariance Matrix**
Input	Accelerometer	$\vec{w}_{a\{b\}}$	$diag\{.001,.001,.001\}\frac{m^2}{s^4}$
	Gyro	$\vec{w}_{\omega\{b\}}$	$diag\{.00005,.00005,.00005\}\frac{rad^2}{s^2}$
	Target Random Walk	$\vec{w}_{t\{i\}}$	$diag\{10,10,10\}\frac{m^2}{s^4}$
Output	GPS Position	$\vec{v}_{p_a\{i\}}$	$diag\{.4,.4,1\}m^2$
	GPS Velocity	$\vec{v}_{\dot{p}_a\{i\}}$	$diag\{.003,.003,.003\}\frac{m^2}{s^2}$
	Pixel	$\vec{v}_{i_f\{b\}}$	$diag\{.00001,.00001\}m^2$
	Target Measurement	$\vec{v}_{p_t\{i\}}$	*variable*

The next step involves defining a relevant simulation environment. This consists of determining the initial states for the aircraft, features, and target and their relationships as a function of time. These conditions are illustrated in the two diagrams of Fig. 7-2, which are not drawn to scale. In (a), the top view of the environment is shown while (b) depicts the side view from an observer looking West. The coordinate frames only represent direction and do not define the inertial coordinate system origin. The UAV begins flight in a Northwest direction along the dotted arrow with a nominal velocity of 44.704m/s and altitude 100m above flat ground, defined by features. The slight initial Northwesterly jog in the flight plan allows for filter and aircraft state excitement. At 750m to the North of the aircraft's initial position, the ground terrain begins sloping downward at approximately 20°. The target's initial position is located at 550m to North of the aircraft's initial position. After nearly 15 seconds of flight, the target, heading due North along the dotted arrow with the same nominal velocity as the aircraft, comes into view. After the target travels 200m to the North of its starting position, it turns and follows the slope maintaining its same velocity. The slope is included to verify that the target can be tracked in 3D terrain. The features are placed in a symmetric triangular arrangement, which allows easy reference for use in the target measurement projection equations. Also, more features are present in the actual simulation than shown in the diagrams. These features are dropped from the diagram to avoid clutter.

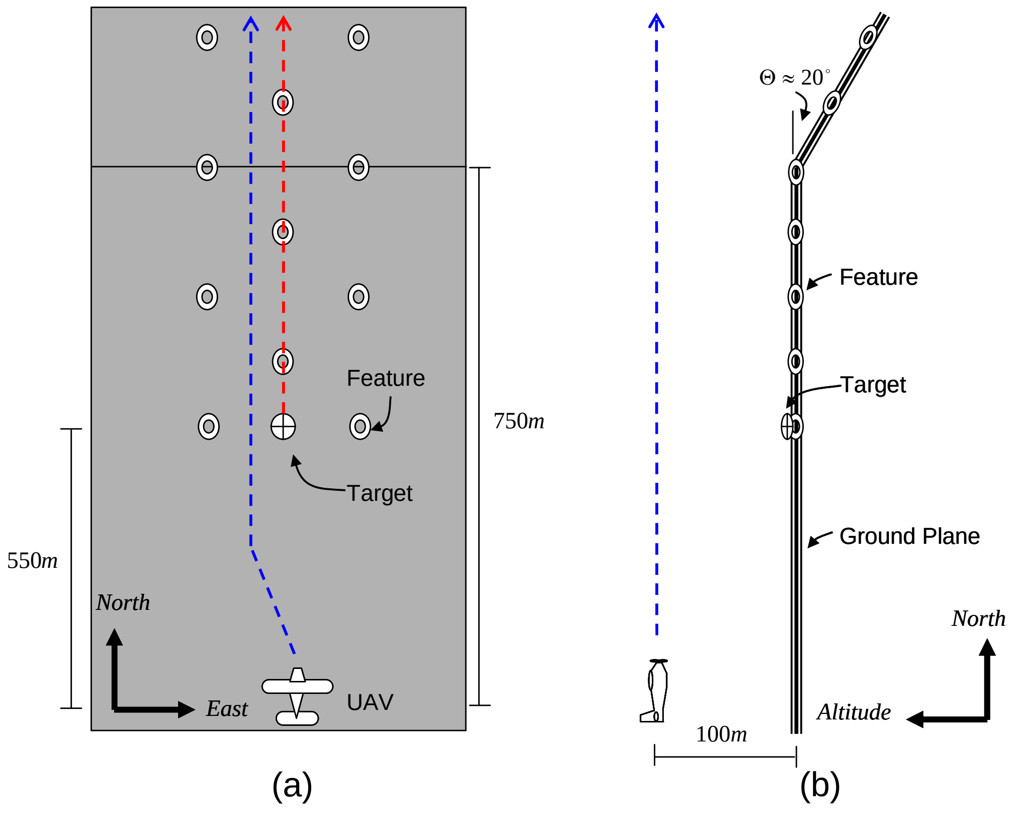


Figure 7-2: Simulation Environment (a) Top View and (b) Side View

Once the initial states are determined, the filters are initialized with mean estimates that are perturbed from their true values by amounts consistent with the initial error covariance matrices. This is accomplished with (4.53) and (4.54) where the initial covariance matrix is used and only one set of random points is generated. While initializing features in this manner is statistically consistent, the true feature locations are not available in application. Therefore, feature mean and covariance initialization techniques are the topic of the next chapter.

The UKF SLAM algorithm, detailed in section 4.3.2, commences at the start of the simulation. The only modification required for the algorithm includes quaternion normalization after (4.65) and (4.80).

$$\bar{q} = \frac{\begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T}{\left\| \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} \right\|} \tag{7.1}$$

Normalization is required due to integration and UKF correction errors. Ref [5] outlines a more complex method for insuring the vector is normalized within the filter algorithm. For the first several seconds of implementation, no features are visible and the UKF SLAM estimator only generates a GPS/INS. Once features are observed their states are added to the state vector, until they are no longer observed. The loosely coupled target KF also begins estimation only after the target is observed.

7.2 Feature Updating

The tremendous quantity of features defining an environment must be managed effectively to provide reasonable computation load, a requirement for real-time applications. Finding a practical solution to the problem is a major topic receiving substantial attention from SLAM researchers. The authors of Ref. [16] present a compressed filter. This filter is ideal for applications in which a vehicle enters an environment where many of the estimated features are temporarily not observed. Once in the new environment, only local feature estimates are updated. A total filter update, based upon the evolution of the state error covariance matrix, is performed once the complete environment is reobserved. Another method for constant-time SLAM involves the creation of submaps. In [17], each submap contains multiple local features, which may also be members of other submaps. At any given time, the vehicle is located within one active submap, and SLAM is performed on the vehicle and local feature states within this map. The global estimates for feature locations are improved through a map location estimation process which takes place when the vehicle moves to a new submap. The process requires the redefinition of the submap root location to be coincident with the feature location with the lowest uncertainty.

While many methods for increasing SLAM computation efficiency exist, the most feasible alternative for this application involves the dropping of unobserved features. Unfortunately, the deletion of states from a filter results in what is commonly referred to as filter information loss. Since the feature locations and vehicle locations are related through dynamic/observation models and nonzero correlation coefficients within the state covariance matrix, the Kalman gain may modify feature estimates, even when they are not currently observed. The correction also results in the update of the state error covariance. Therefore, the covariance and mean information associated with a feature when it is removed can not be used when the same feature is reobserved. The deletion of a feature's mean states and its associated rows within the state covariance filter is allowed as removal does not affect the statistical consistency of the mapping process [18].

The adding and dropping of features reduces the magnitude of computations, but results in code that is slightly more complex. At the beginning of each correction step of the SLAM UKF algorithm, new image data are generated. Part of image generation involves the determination of which features are viewed in the current image based upon the camera's actual location, orientation, and FOV. The newly observed feature mean values are appended to the filter's mean vector. The initial 3x3 covariance matrix for each feature, defined during environment initialization, is then concatenated to the current state covariance matrix. Features that are no longer in view have their mean values dropped from the filter and the row and column corresponding to their states in the state covariance matrix. The operations specified here obey the developments in section 6.2.

7.3 Results

The error magnitude, or Euclidean norm, for a feature position estimate is shown in Fig. 7-3. During the first ten seconds of flight, this feature is not in view. Once in view, the UKF dynamically updates and improves the position estimate of the feature. Other features exhibit similar behavior.

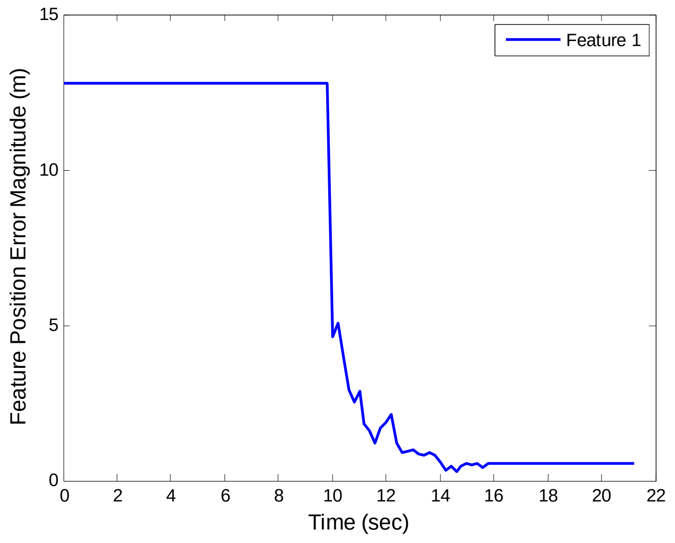


Figure 7-3: Feature One Position Error Magnitude

The next plot emphasizes the orientation accuracy for the aircraft gained through SLAM estimation as compared to traditional GPS/INS estimation. Improved orientation estimates are particularly noticeable when the features come into view after approximately nine seconds. This behavior is expected because the observation model for the features depends heavily upon the position and orientation of the aircraft. In fact, the EKF GPS/INS actually developed a substantial orientation bias as the aircraft started entering steady level flight.

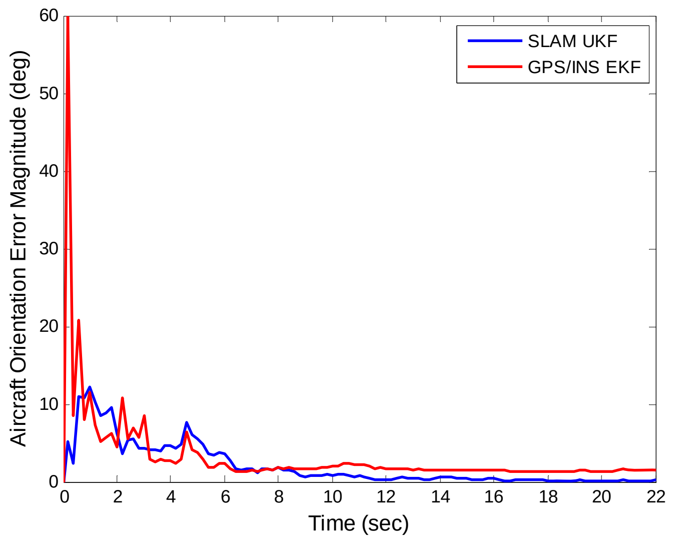


Figure 7-4: Aircraft Orientation Error Magnitude

The target position error illustrates the effectiveness of the target tracking technique developed. Fig. 7-5 plots the error magnitude as a function of the time during which the target is observed. The error peak between four and five seconds corresponds to the point where the target vehicle turns and transverses down the slope. While the error never converges completely to zero, the target state estimates do track the target.

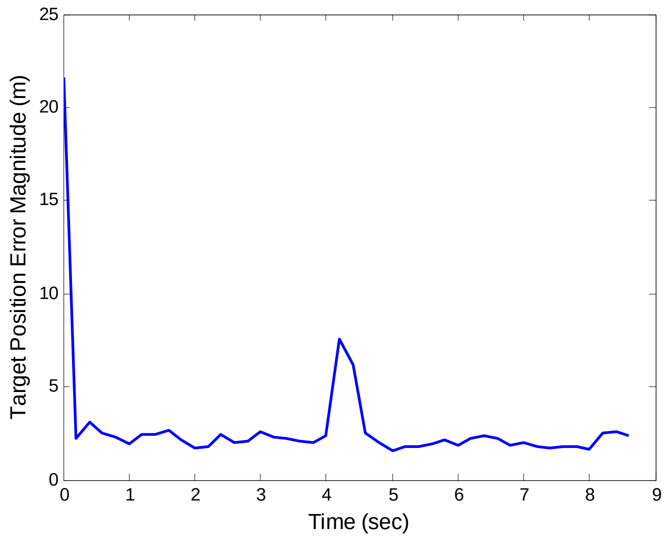


Figure 7-5: Target Position Error Magnitude

The target velocity error magnitude is shown in Fig. 7-6. The general shape of this plot is very similar to the target position error plot. However, the error peak which occurs between four and five seconds is significantly larger. This is expected as no direct observation measurement for the target's velocity is provided, and the zero mean random walk on velocity implies that the average acceleration should be zero.

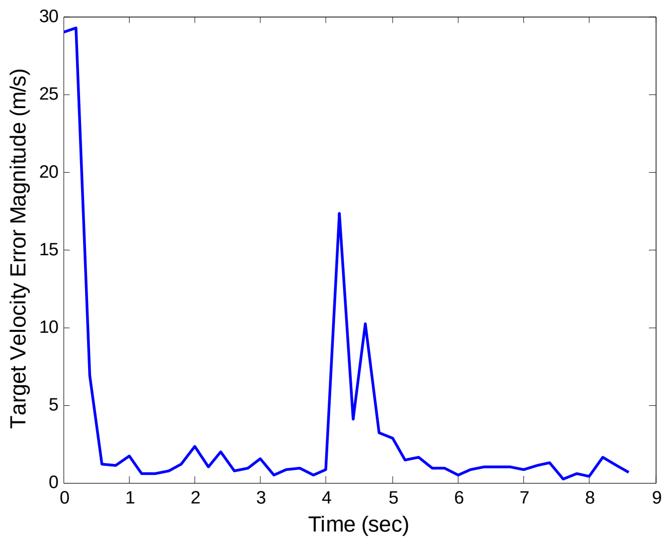


Figure 7-6: Target Velocity Error Magnitude

Overall, these error plots confirm that the estimator setup developed is capable of tracking the states of a target, contingent upon the accurate estimation of local terrain features. The information gained from this estimation technique may be used to provide tracking information for a range of applications.

Chapter 8: Feature Initialization

While the feature mean and covariance matrix initialization performed in the previous chapter is statistically consistent, it required knowledge about the true 3D location of the features. This information is not available with experimental data. The only sensor data available from the features are the pixel measurements returned from the feature extraction algorithm discussed in the next chapter. The problem now becomes one of constructing a 3D feature position initialization from the sensor data and camera pose estimates. Unfortunately, the camera states are plagued with uncertainty and are typically biased over short intervals as the estimate error is generally not a white signal. Therefore, the initial covariance matrix for each feature's position must reflect the transformation of these uncertainties into an accurate 3x3 feature covariance matrix. Also, the set of equations which determine the mean value must themselves be designed to handle perturbation from the true camera pose and pixel locations. These difficulties have led to the implementation and comparison of three different initialization techniques: single triangulation, ground plane projection, and depth conditioning.

After the equation development for each of the initialization techniques is introduced, simulation is used as an identical and reproducible medium for feature mean and covariance analysis. This is accomplished by setting the seed values for the SIMULINK and image simulation function random number generators to a known value for every simulation. The same features are also used for all mean comparisons plots in the next three analysis sections. These 50 features are initially randomly distributed in a 400m x 200m plane located 100m below the aircraft. The feature locations are also initially perturbed about the plane in the z direction by random values between +/- 20m. The metric used for mean evaluation is the average magnitude of the position error for all of the features. This metric is used for both the initial and final positions. For all but the single triangulation method, both a tightly coupled SLAM estimator and a loosely coupled estimator are compared. Here LC refers to the separation of feature and aircraft states into two separate estimators as shown in Fig. 8-20. This comparison is needed because detrimental feature mean and covariance initial estimates adversely affect the aircraft state estimates in tightly coupled SLAM. The accuracy of the initial covariance estimate values are also examined with the LC filter setup. This is accomplished by determining the error in the x, y, and z axes and transforming this value into a standard deviation based upon the initial covariance matrix. However, determining the effectiveness of this parameter is slightly more difficult as the covariance matrix provides statistical information based upon several samples. Therefore, 100 total features are created and used for the covariance analysis where the percentage of initial mean estimates within the standard confidence intervals is plotted.

8.1 Single Triangulation

The initialization equations developed for the single triangulation are based upon the environment shown in Fig. 8-1. In the diagram, the measured pixel coordinates from two poses are known along with the corresponding camera pose information. The “1” and “2” subscripts and superscripts are used as a reference to the two poses. Since the camera poses and measurements contain uncertainty, a high likelihood exists that the projected lines from the camera pose body frame through the pixel measurements are skew. Therefore, the assumption is made that the feature lies halfway along the shortest line between the two projected lines.

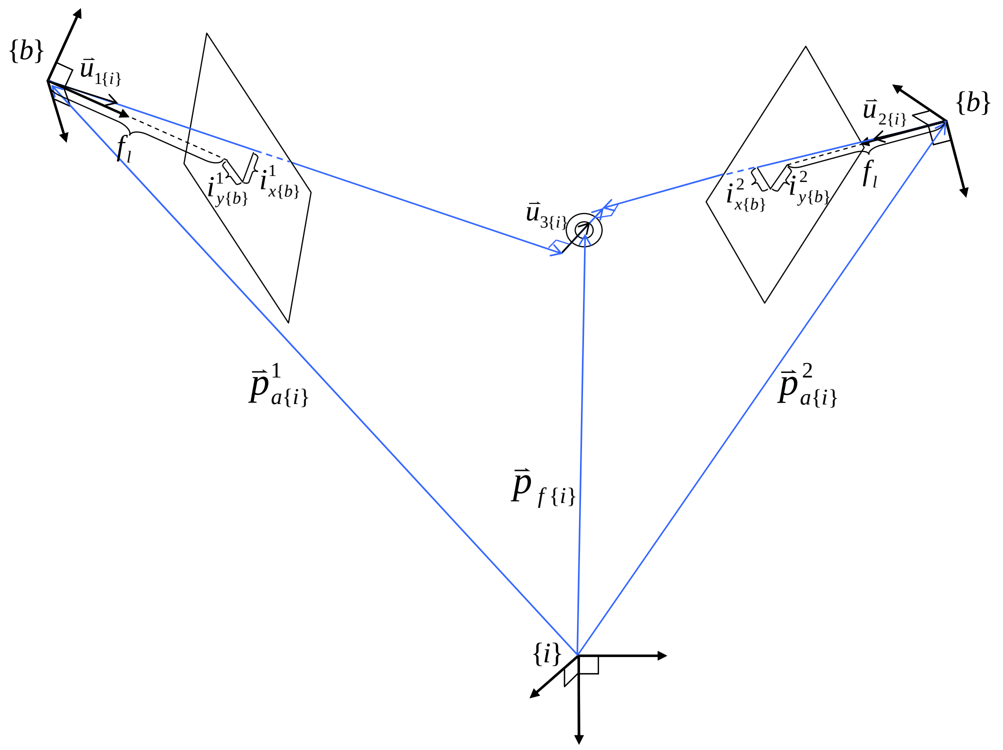


Figure 8-1: Single Triangulation

The first step in acquiring the feature's position involves determining the unit vectors, $\vec{u}_{1\{i\}}$ and $\vec{u}_{2\{i\}}$, along which the feature should reside.

$$\vec{u}_{1\{i\}} = \mathbf{R}_{\{ib\}} \frac{[i^1_{x\{b\}} \quad i^1_{y\{b\}} \quad f_l]^T}{\left\|[i^1_{x\{b\}} \quad i^1_{y\{b\}} \quad f_l]\right\|} \tag{8.1}$$

$$\vec{u}_{2\{i\}} = \mathbf{R}_{\{ib\}} \frac{[i^2_{x\{b\}} \quad i^2_{y\{b\}} \quad f_l]^T}{\left\|[i^2_{x\{b\}} \quad i^2_{y\{b\}} \quad f_l]\right\|} \tag{8.2}$$

$\vec{u}_{3\{i\}}$, which is orthogonal to $\vec{u}_{1\{i\}}$ and $\vec{u}_{2\{i\}}$, is found by normalizing the cross product of $\vec{u}_{1\{i\}}$ and $\vec{u}_{2\{i\}}$.

$$\vec{u}_{3\{i\}} = \frac{\vec{u}_{1\{i\}} \times \vec{u}_{2\{i\}}}{\left\|\vec{u}_{1\{i\}} \times \vec{u}_{2\{i\}}\right\|} \tag{8.3}$$

The following vector equality may be written.

$$\vec{p}^{\,1}_{a\{i\}} + \lambda_1 \vec{u}_{1\{i\}} + \lambda_3 \vec{u}_{3\{i\}} = \vec{p}^{\,2}_{a\{i\}} + \lambda_2 \vec{u}_2 \tag{8.4}$$

Here λ_1, λ_2, and λ_3 are unknown scalar values. Manipulation of this equation results in

$$\begin{bmatrix} \vec{u}_{1\{i\}} & -\vec{u}_{2\{i\}} & \vec{u}_{3\{i\}} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \vec{p}^{\,2}_{a\{i\}} - \vec{p}^{\,1}_{a\{i\}} \tag{8.5}$$

where the solution is given by

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \vec{u}_{1\{i\}} & -\vec{u}_{2\{i\}} & \vec{u}_{3\{i\}} \end{bmatrix}^{-1} \begin{bmatrix} \vec{p}^{\,2}_{a\{i\}} - \vec{p}^{\,1}_{a\{i\}} \end{bmatrix} \tag{8.6}$$

provided the matrix of unit vectors is not singular. As long as $\vec{u}_{1\{i\}}$ and $\vec{u}_{2\{i\}}$ are not parallel, this matrix will have full rank. When the unit vectors are parallel an infinite number of values for λ_1 and λ_2 exist that solve (8.4). Even before the unit vectors become parallel, the matrix starts to become ill-conditioned. For this reason, an angle threshold condition, discussed below, must be passed before a feature is initialized. The initial feature location, $\vec{p}_f$, is then assumed to reside halfway between the skew projected vectors along $\vec{u}_{3\{i\}}$.

$$\vec{p}_f = \vec{p}^{\,1}_{a\{i\}} + \lambda_1 \vec{u}_{1\{i\}} + \frac{\lambda_3}{2} \vec{u}_{3\{i\}} \tag{8.7}$$

While (8.7) provides a nonlinear equation for determining the feature location from two poses, the covariance matrix for this feature is still needed. Due to the highly nonlinear nature of the equation, the UT is employed for mean and covariance calculation of a feature's initial position. As mentioned, the UT provides a superior mean and covariance estimate for nonlinear equations when compared to Jacobian based covariance techniques.

8.1.1 Analysis

In general, the integrity of this initialization depends on the stereoscopic nature of the camera poses. In other words, by making the angle between the two legs of the triangle larger, the resulting triangulation should be more accurate. This is especially true for systems, like this SLAM UKF estimator, where knowledge of the true pose of the camera is an estimate. Therefore, the angle between the two unit vectors, $\vec{u}_{1\{i\}}$ and $\vec{u}_{2\{i\}}$, is calculated and then compared with a minimum angle threshold. If the angle is greater than the threshold, the feature is allowed to be initialized with the single triangulation equations. The angle, α, between the vectors given by (8.1) and (8.2) is determined by taking the arccosine of the dot product.

$$\alpha = a\cos\left(\vec{u}_{1\{i\}}^{T}\vec{u}_{2\{i\}}\right) \tag{8.8}$$

Simulation results confirm the argument that a greater initial angle threshold results in better initial feature positions. In order to generate data for Fig. 8-2 and Fig. 8-3, the only variable that is allowed to change is the angle threshold. As the angle threshold increases, the average error decreases as expected. This same general trend is illustrated for the final position error as shown in Fig. 8-3. Unfortunately, the LC filter estimates diverged as denoted by the high final error values. The SLAM filter estimates also diverged. However, in a tightly coupled filter, this results in diverging aircraft state estimates. Therefore, no discernable information is available from either of the SLAM filter plots, and they are not shown here.

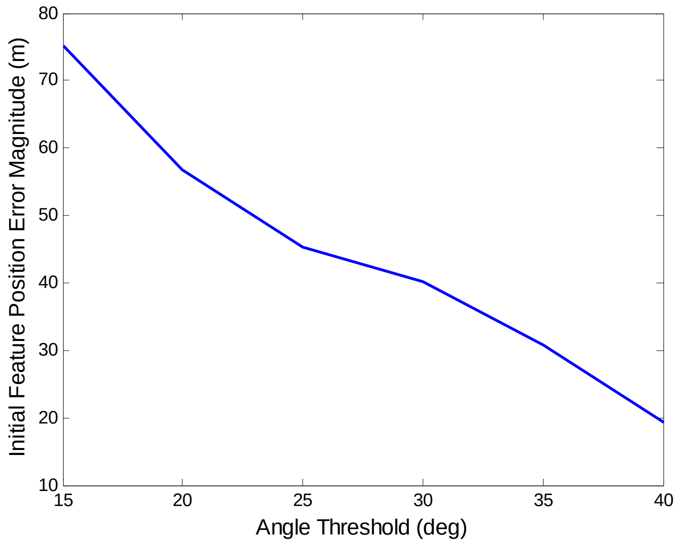


Figure 8-2: LC Single Triangulation Feature Initial Position Error Magnitude Plot

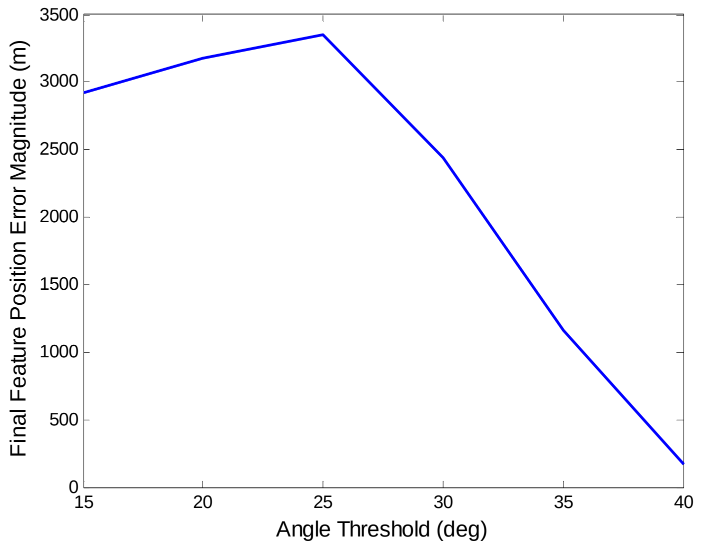


Figure 8-3: LC Single Triangulation Feature Final Position Error Magnitude Plot

A covariance matrix analysis is used to determine if the generated initial covariance matrix captures the uncertainty in the initial feature position. This analysis uses 100 features that are initialized after passing the angle threshold requirement of 35deg. Calculations reveal an average 21m initial feature position error. While the average initial covariance matrix is very large (det = $3.060*10^5 m^2$), Fig. 8-4 shows the UT only slightly overestimates each covariance matrix for each of the x, y, and z axes. Ideally, for a normally distributed variable, the one sigma value should reside at the 68.2% confidence interval, while the two, and three sigma values should denote the 95.5%, and 99.7% intervals, respectively. Although the initial mean error is large, the large covariance matrix enables the filter to perform properly. The high covariance terms allow the estimates greater freedom of movement because their uncertainty is large. However, once a feature's mean estimate approaches the singularity plane defined in section 3.2.2.2, the filter's performance degrades and eventually diverges. A possible solution to this problem might be to increase the angle threshold. Unfortunately, this decreases the number of correction updates since each feature is in view for a finite number of frames. Clearly, other methods are needed to address the deficiencies of a single triangulation initialization.

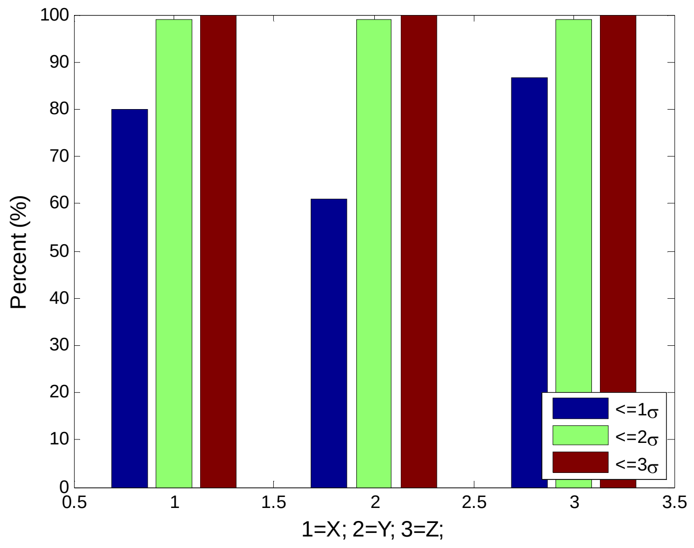


Figure 8-4: LC Single Triangulation Initial Feature Covariance Analysis

8.2 Ground Plane Projection

In many scenarios, the local terrain along which a ground target may travel is not very rugged. For these cases, the local features that define the terrain should reside very near a ground plane. Therefore, initializing the features about a ground plane might provide a good initial estimate. This ground plane projection scheme is illustrated in Fig. 8-5.

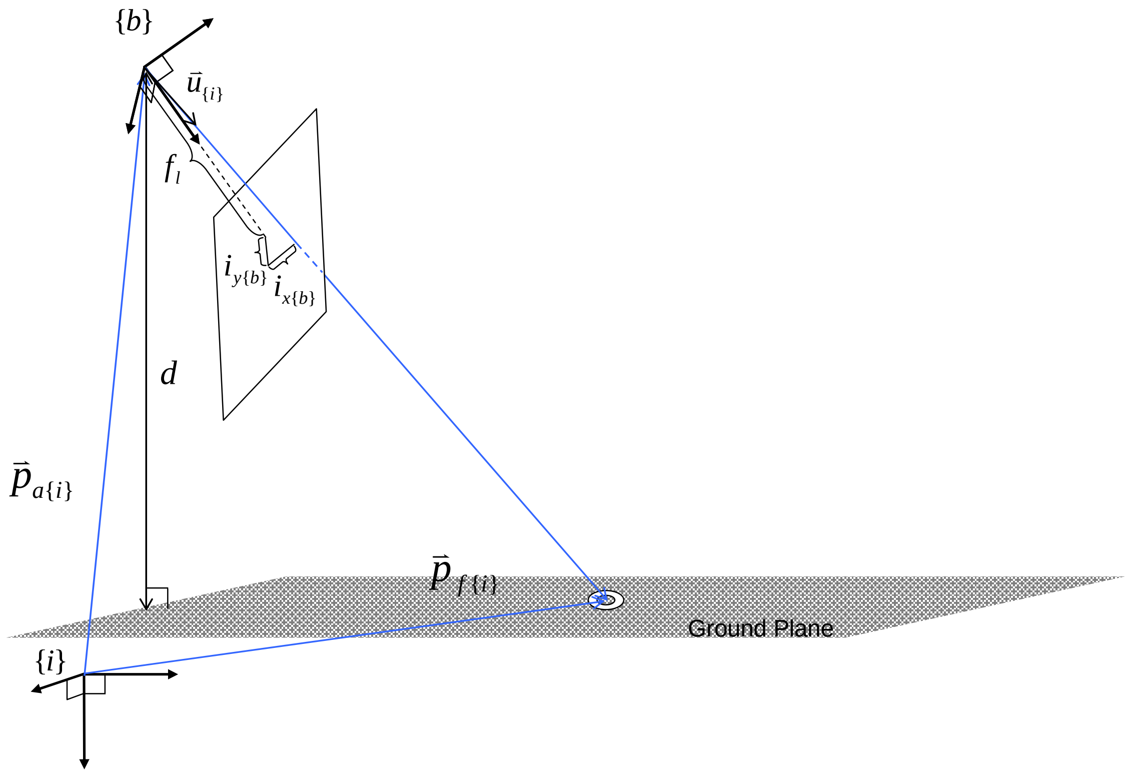


Figure 8-5: Ground Plane Projection

The scale factor, λ, along the unit vector, $\vec{u}_{\{i\}}$, is found by the following equation

$$\lambda = \frac{d}{\begin{bmatrix}0 & 0 & 1\end{bmatrix}\vec{u}_{\{i\}}} \tag{8.9}$$

where d is a scalar distance to the ground plane from the aircraft and the unit vector is given by (8.1). If the aircraft flies at a steady nominal altitude, then the scalar d is constant. If the aircraft constantly changes altitude, this scalar value is determined from the z component of the aircraft's position vector and the altitude of the ground plane with respect to the inertial frame. The feature's initial position, $\vec{p}_{f\{i\}}$, is then given by

$$\vec{p}_{f\{i\}} = \vec{p}_{a\{i\}} + \lambda\vec{u}_{\{i\}}\,. \tag{8.10}$$

Equation (8.10) is used by the UT algorithm to capture the current feature position and covariance. Most of the mean values are provided by the SLAM UKF estimator and the pixel coordinate measurement vector. However, the properties assigned to the distance variable d are available for tuning. In certain cases, an above ground level altitude sensor may provide satisfactory measurements of this variable. Otherwise, the ground plane location should be chosen based upon prior knowledge of average terrain elevation and the flight altitude of the aircraft. The variance of the distance to the plane is also important. If the variable is chosen to have zero variance, the UT interprets this as a constraint in the feature's z direction. In other words, the feature is initialized in the plane with a covariance that includes zero values for all terms associated with the down direction. To avoid this artificial constraint, the variance on the distance variable is tuned such that the features are not fixed to the ground plane, but are allowed some movement.

8.2.1 Analysis

Perhaps the greatest benefit from the ground plane projection technique is the ability to initialize features the first time they are viewed. This allows successive frames to be used strictly for correction updates. For non-gimbaled cameras mounted on UAV's, every observation is critical since the number of frames in which a feature is viewed tends to decrease greatly as the aircraft velocity increases and/or the altitude decreases.

The greatest drawback of this method originates from the value chosen for the distance to the ground plane and its associated variance. Underestimation of the variance of this distance results in feature position estimates that stubbornly reject convergence to their true locations. Overestimation results in feature estimates that require more observations for convergence and that may erratically shift about their true locations during their first few observations. Incorrect distance estimates result in even more negative behavior. These results are illustrated in Fig. 8.6 through Fig. 8.9 where the average feature error is plotted as a function of nominal ground plane distance and the variance associated with the distance. In general, as the variance of the distance variable decreases, the overall error tends to increase. This characteristic is especially true for the SLAM estimator when the distance to the actual ground plane, located at approximately 100m below the aircraft, is incorrect. While the final error is relatively small when the correct nominal ground plane distance is used, the SLAM estimator is not very robust to incorrect altitude and variance estimates. This is expected as the incorrect initial feature positions are correlated to the aircraft states in tightly coupled estimators. On the other hand, the LC estimator appears to be less sensitive to incorrect aircraft to ground plane distance estimates. In fact, the initial error plot confirms this statement as every combination of variance results in the same initial error. The aircraft estimate is not influenced by the mapping of features in the LC estimator. Another interesting LC estimator trait is illustrated in Fig. 8.9. As the variance increases, the final feature error tends to find a minimum when the nominal ground plane distance is larger than the true nominal ground plane distance, even after very large incorrect initial estimates. As the distance between the aircraft and the expected ground plane increases, the covariance matrix values also tend to increase. For example, a small perturbation in orientation results in a larger change in the ground plane feature position when the ground plane distance from the aircraft increases. A larger covariance matrix corresponds to a higher uncertainty in the initial feature estimate, and thus the estimate is allowed greater freedom of movement during future corrections.

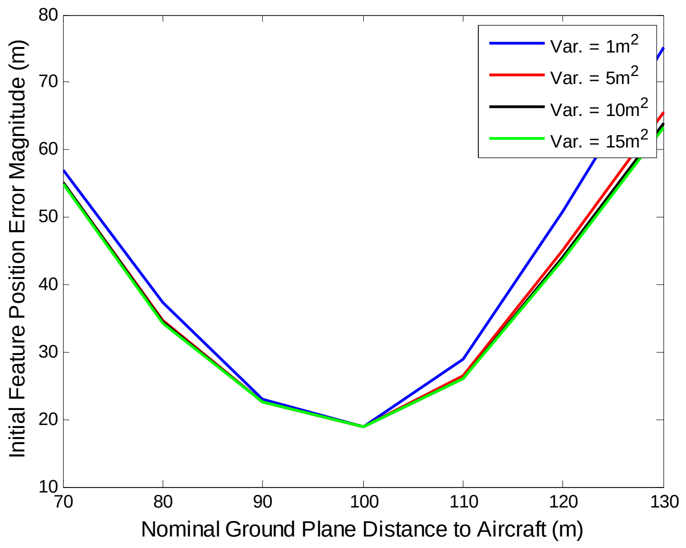


Figure 8-6: SLAM Ground Plane Projection Feature Initial Position Error Magnitude Plot

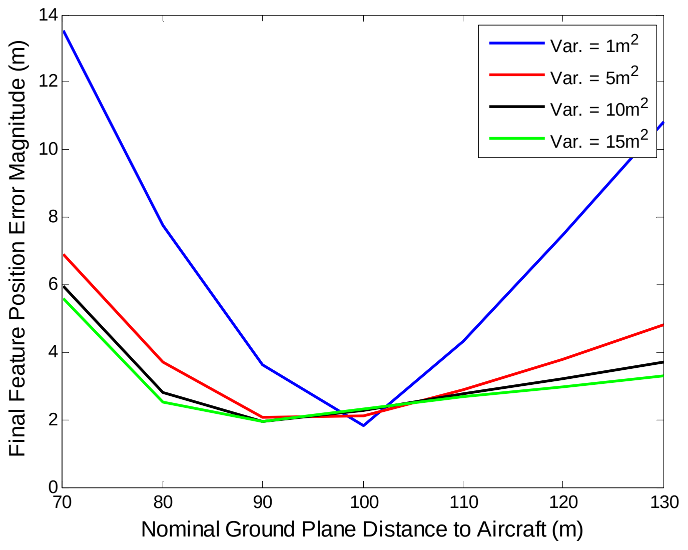


Figure 8-7: SLAM Ground Plane Projection Feature Final Position Error Magnitude Plot

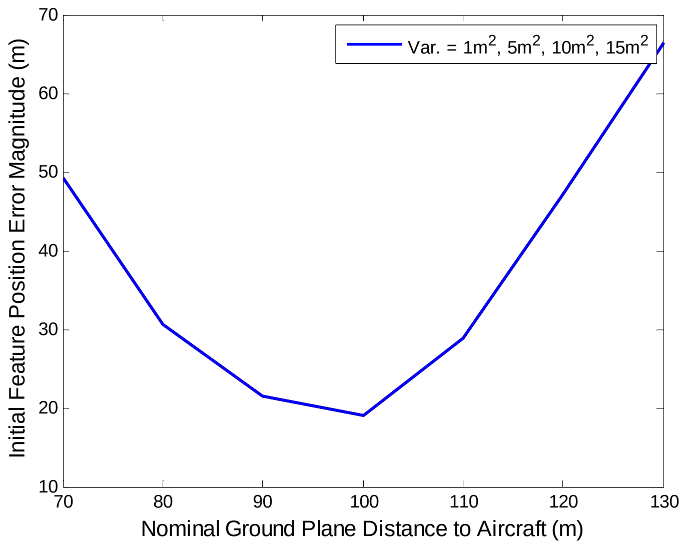


Figure 8-8: LC Ground Plane Projection Feature Initial Position Error Magnitude Plot

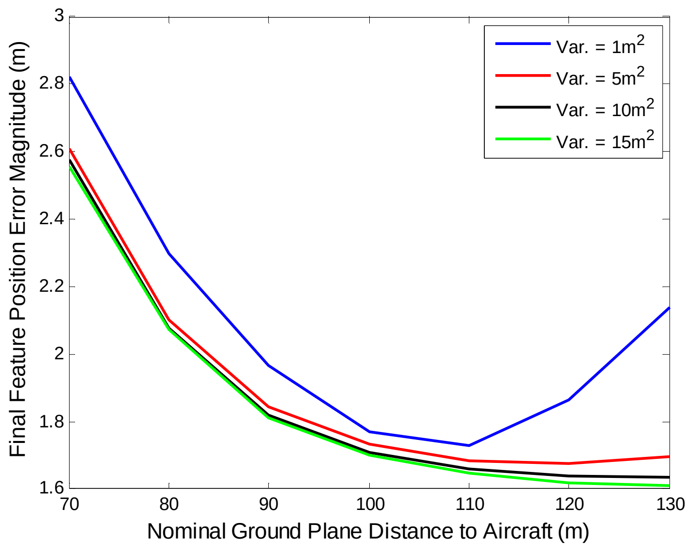


Figure 8-9: LC Ground Plane Projection Feature Final Error Magnitude Plot

The covariance analysis data is collected from a simulation utilizing the ground plane projection equations. A 100m ground plane distance is used with a 10m^2 variance. As can be readily inferred from Fig. 8-10, the ground plane projection severely underestimates the true covariance for all three axes. In fact, this initialization technique produces an average initial position error of 20.3m, equivalent to the single triangulation initial error, with a relatively small valued average covariance matrix (det = $2.612*10^3$m^2). Even with this underestimation, the estimator is able to provide fairly accurate final position estimates.

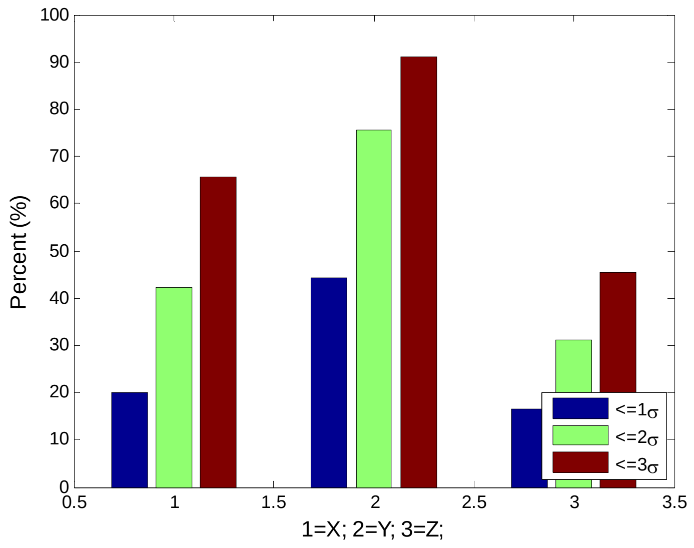


Figure 8-10: LC Ground Plane Projection Initial Feature Covariance Analysis

8.3 Depth Conditioning

Literature searches yielded the final initialization technique examined. In their MonoSLAM research, Davison et. al. encountered a similar initialization dilemma where the depth to a feature is not directly obtainable from the observation model [12]. To overcome this difficulty, they start by generating a semi-infinite 3D line in space along which the feature should reside based upon the camera's states and the feature's pixel measurements. In general, this line is described by unit vector in the inertial frame that originates from the end of the camera position vector. The depth along the line in which the feature is located, however, is unknown. A set of particles are then uniformly distributed along the line. Each particle represents a new scalar magnitude, λ, or depth along the line. Each successive image then provides an observation of the measured feature's pixel coordinates. All the particles are then projected into the new image space where their likelihoods are determined for Bayesian re-weighting of the particle distribution. The likelihoods consider the uncertainty due to the camera's location at the given time, pixel measurement noise, and also uncertainty in the parameters of the 3D line in space along which the features should reside. As the number of measurements increase, the depth distribution becomes more Gaussian. Once a certain ratio of the standard deviation over mean depth of particles is reached, the feature is initialized.

A modification of Davison's approach as applied to UAV mounted monocular vision systems is illustrated in Fig. 8-11. Using the initial image of a feature and the current camera states, a unit vector defining the line along which the feature is assumed to lie is formed. The set of depth hypotheses, or particles, are then uniformly distributed along this line. The particles are shown as circles in the figure. Three parameters regarding the initial particles are available for fine-tuning and adjustment in the algorithm. These include the number, mean, λ_m; and spread, λ_s, of the particles.

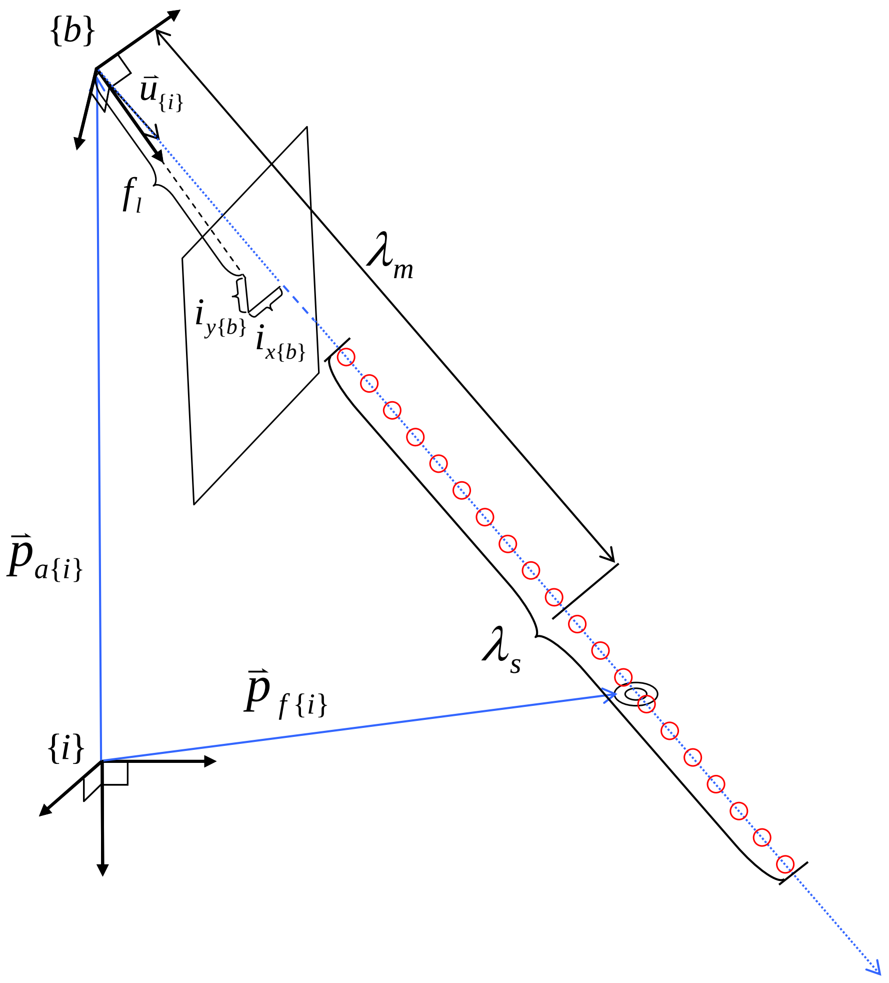


Figure 8-11: Depth Conditioning

At each new observation of the feature, the conditional relative likelihood of each particle is determined. This involves the propagation of all of the particles through the nonlinear camera observation model given by (3.21) and (3.22). Assuming the measurement noise has a Gaussian

PDF, the likelihood, q_i, that the "ith" particle's projection into image space is equal to current measurement must be proportional to

$$\frac{1}{(2\pi|\mathbf{R}|^{.5})}\exp\left(\frac{-[\vec{e}_i]^T\mathbf{R}^{-1}[\vec{e}_i]}{2}\right) \tag{8.11}$$

where

$$\vec{e}_i = \vec{i}_{fm\{b\}} - \vec{i}_{fi\{b\}} . \tag{8.12}$$

$\vec{e}_i$ is the error between the current pixel measurement vector, $\vec{i}_{fm\{b\}}$, and the image space representation of the "ith" particle, $\vec{i}_{fi\{b\}}$. $\mathbf{R}$ is usually the 2x2 measurement noise covariance matrix. However, since the observation model is also a function of the current orientation and position of the camera, uncertainty in those parameters must also be ascertained. Therefore, the total covariance is given by

$$\mathbf{R} = E\left\{(\vec{v}_k)(\vec{v}_k)^T\right\} + \left(\left.\frac{\partial(\vec{h}(\vec{x}))}{\partial\vec{x}}\right|_{\vec{x}=\bar{\vec{x}}}\right)\mathbf{P}^{xx}\left(\left.\frac{\partial(\vec{h}(\vec{x}))}{\partial\vec{x}}\right|_{\vec{x}=\bar{\vec{x}}}\right)^T \tag{8.13}$$

where $\vec{v}_k$ is the zero mean Gaussian pixel measurement noise, and the current state estimate vector is

$$\vec{x} = \begin{bmatrix} \vec{p}_a \\ \vec{q} \end{bmatrix} . \tag{8.14}$$

The covariance matrix, $\mathbf{P}^{xx}$, of $\vec{x}$ is found via the linear covariance transformation properties outlined in section 6.2. The relative likelihood for each of the particles are determined by normalizing all q_i. A set of new particles are randomly generated based upon these new relative likelihoods [15]. This process is completed for each additional observation of the feature, upon which the depth distribution becomes increasingly Gaussian. Fig 8-12 illustrates this progression. The data generated for these plots are obtained from the Navion simulation where 200 particles were used with a mean value of 150m and a spread of 200m. At each discrete time step, the particle data for one of the features was stored. The first row of plots contain the ray along which the feature should reside based upon the initial camera states and pixel measurements for the feature. They also contain the particle locations denoted by the circles along this line. The camera's location is given by the body axes plots of the aircraft at (0m,100m). Histogram plots of the lambda depth hypotheses fill in the second row. Each column corresponds to a different view number, as given in the caption. Even though all views of this particle are not shown for brevity, a general trend is obvious. The particles tend to become more clustered with increasing number of views. The distribution also appears to take on the tradition bell curve shape of a Gaussian PDF.

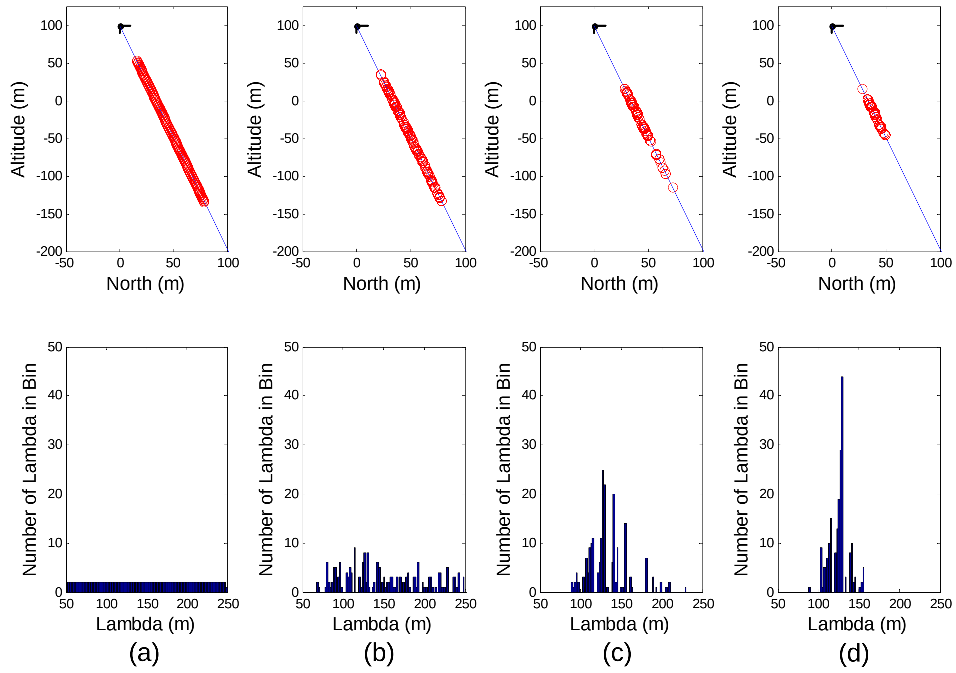


Figure 8-12: Particle Location and Histogram for a Feature after (a) One View, (b) Three Views, (c) Five Views, and (d) Seven Views

For initialization in this research, the ratio of the variance over depth must be below a preset threshold. This is slightly different than Davison et al. who use the standard deviation over depth ratio as the threshold condition [12]. Both the mean and variance are numerically calculated after each re-sampling. Given a ratio threshold of .25m, the feature in the above example initialized with a final mean lambda value of 117.1850m as opposed to the actual value of 117.2857m. While this mean value is very close to the actual, the initial ray in space along which the feature should reside is not completely accurate because it was obtained with estimates and measurements. The corresponding position initialization has a slightly greater error.

Finally, once a feature is ready for initialization, its new mean position and covariance matrix are determined. This is accomplished by the UT. The position of the feature is given by

$$\vec{p}_{f\{i\}} = \vec{p}_{a\{i\}} + \overline{\lambda}\vec{u}_{\{i\}} \tag{8.15}$$

where $\overline{\lambda}$ refers to the final depth conditioning mean lambda value, not to be confused with the initial lambda mean of the uniformly distributed particles. The UT considers the uncertainty in the aircraft position, unit vector formulation, and lambda for the determination of the feature's initial mean and covariance matrix.

8.3.1 Analysis

Many variables influence the behavior of the depth conditioning method. Unfortunately, testing the performance of every possible combination of these variables is intractable. For this reason, a standard set of variable values based upon preliminary observations are chosen as a suitable operating point. These values are given as 200 particles with a lambda mean and spread of 200m. In each subsequent test, one variable changes as all others are held constant.

Quantitatively, varying the number of particles did not provide any conclusive results. In general, the final average error varied randomly between 1.5m and 2m as the number of particles varied from 150 to 250. However, lowering the number of particles below 150 did have consequences. During re-sampling, the new distribution of particles is extracted from the previous set of particles. Therefore, the possibility that all of the particles converge to a single point increases as the distance between the initial particles becomes more coarse. This is commonly referred to as sample impoverishment, and several methods exists for overcoming this limitation other than increasing the number of particles [15]. However, this problem never occurred when more than 150 points were used. For this reason and to limit computational loads, 200 points are used during depth conditioning implementations.

Changes to both the λ_m and λ_s parameters slightly affect the final error statistics. Once again, a simulation provides the data necessary for performance assessment. The final results for three different λ_s values are plotted in Fig. 8-13 through Fig. 8-16. With a λ_m value ranging from 150 to 250, all lambda spread error lines produce nearly consistent results. The overall average final feature position error in this region is generally below 2m. When the lambda mean value is significantly incorrect, the error plots tend to increase. This is more prevalent when λ_s decreases. Essentially, the feature's actual position is out of the range of possible depth hypotheses generated. Since the particles are re-sampled from previous sets, this imposes a limitation. Some sample impoverishment techniques may reduce this tendency. Even so, the particle method provides a robust means to initialize particles as fairly significant changes in λ_m do not result in substantial errors. These beneficial characteristics are even more prevalent in the LC estimator results, where the final position error is less than 2m for all combinations of λ_m and λ_s.

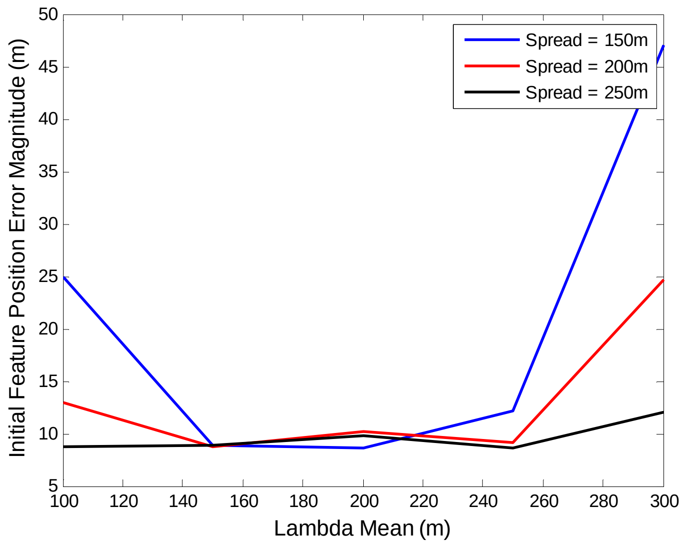


Figure 8-13: SLAM Depth Conditioning Feature Initial Position Error Magnitude Plot

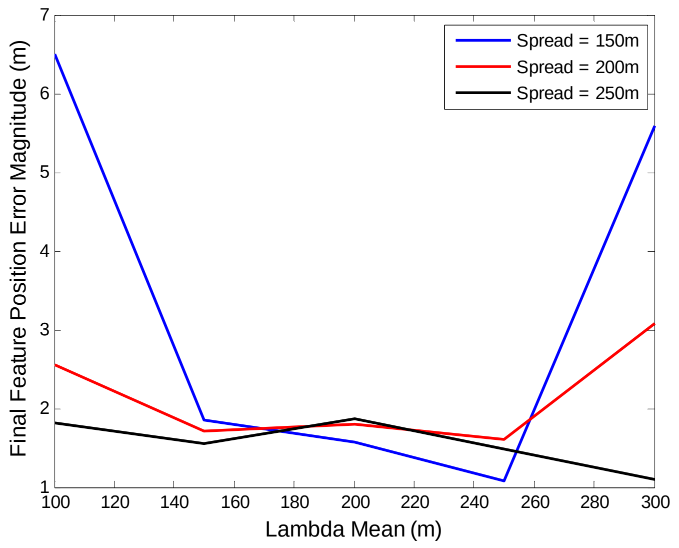


Figure 8-14: SLAM Depth Conditioning Feature Final Position Error Magnitude Plot

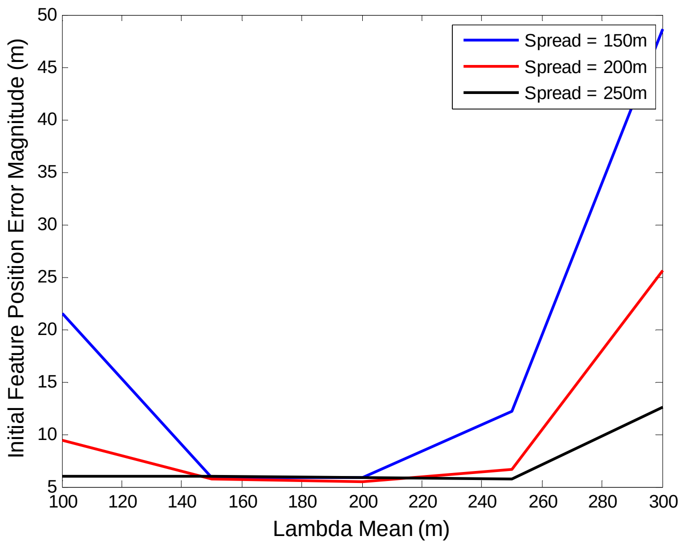


Figure 8-15: LC Depth Conditioning Feature Initial Position Error Magnitude Plot

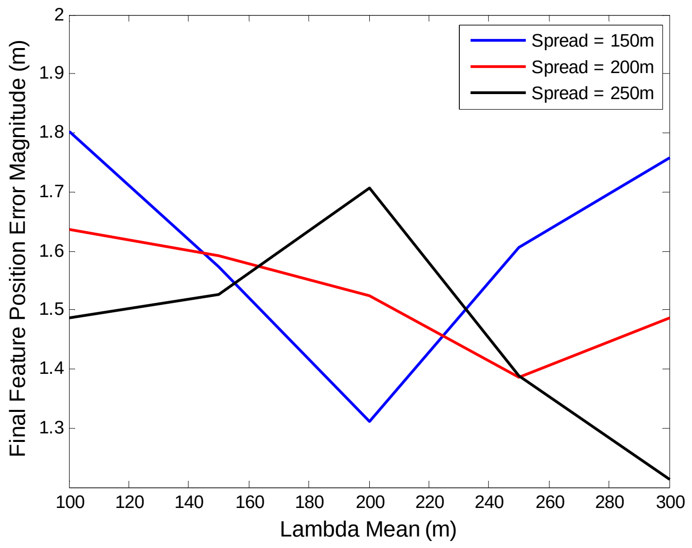


Figure 8-16: LC Depth Conditioning Feature Final Position Error Magnitude Plot

The final check includes the covariance analysis. For this analysis, 200m λ_m and λ_s values are used as they provide an adequate range of depth hypotheses that are not too sparsely distributed. Both the average initial error magnitude (6.71m) and average covariance matrix (det = $1.870*10^3 m^2$) are lower than the previous two initialization techniques. The initial position errors are also accurately described by their corresponding covariance matrices as shown by the data in Fig. 8-17. The combination of accurate initialization and covariance determination strengthen the argument for its use with the experimental data.

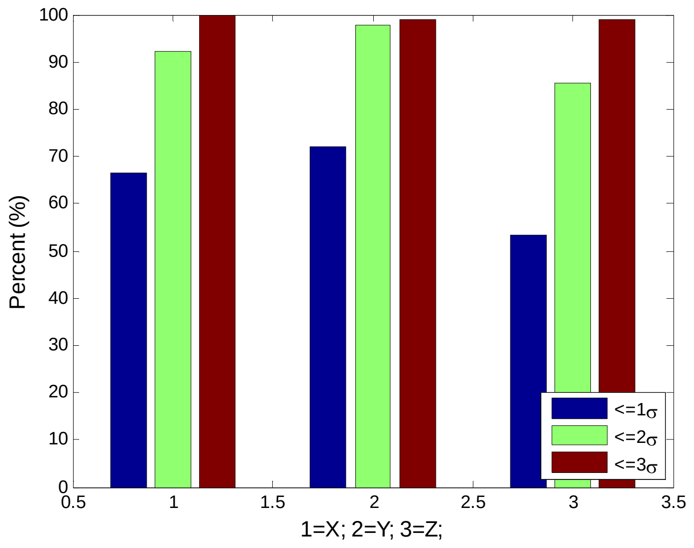


Figure 8-17: LC Depth Conditioning Initial Feature Covariance Analysis

8.4 Initialization Technique Conclusions

Several general trends and conclusions are attainable from the simulation data regarding the mean and covariance initialization techniques. The single triangulation's poor mean estimates and large covariance matrix values promote filter divergence, and it is, therefore, not considered any further. The other two initialization techniques appear to provide fairly similar final error results for the set of test conditions provided. However, other parameters need to be examined including the affects of feature density. Finally, the LC estimator appears to perform better than the tightly coupled SLAM estimator when considering the final feature errors.

Another series of simulations provides data emphasizing the influence of feature density on the ground plane projection and the depth conditioning methods. The LC and SLAM estimation techniques are also compared in this analysis. The simulation environment is identical to that presented in the previous section. However, the number of features comprising the ground plane is allowed to change. Since the volume of the possible feature locations is constant, a higher feature number represents an increase in average feature density. In general the depth conditioning method provides better initial mean estimates for both the SLAM and LC filters as shown in Fig. 8-18.

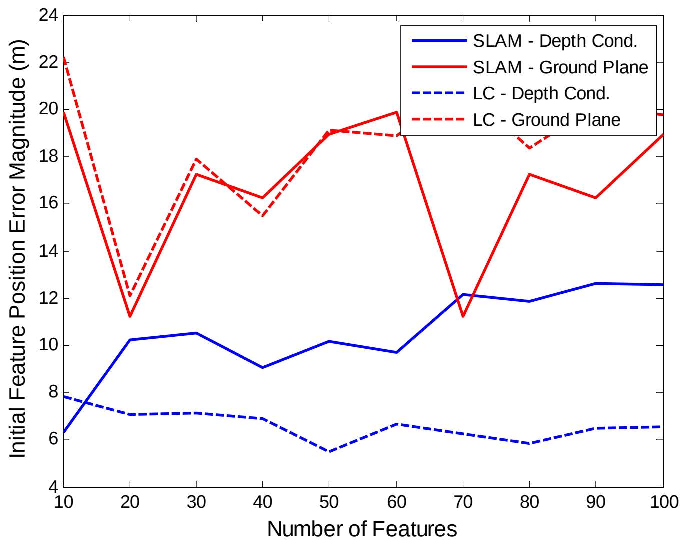


Figure 8-18: SLAM and LC Ground Plane and Depth Conditioning Initial Feature Position Error Magnitude as a Function of Feature Density

In the case of the final feature position RMS error, Fig. 8-19, the general trends are more difficult to ascertain. Generally, the LC estimator average final feature estimates are closer to the true values than the SLAM feature estimates. No statement can be made regarding the effectiveness of the depth conditioning method over the ground plane method or vice versa as their final feature error values are nearly equivalent.

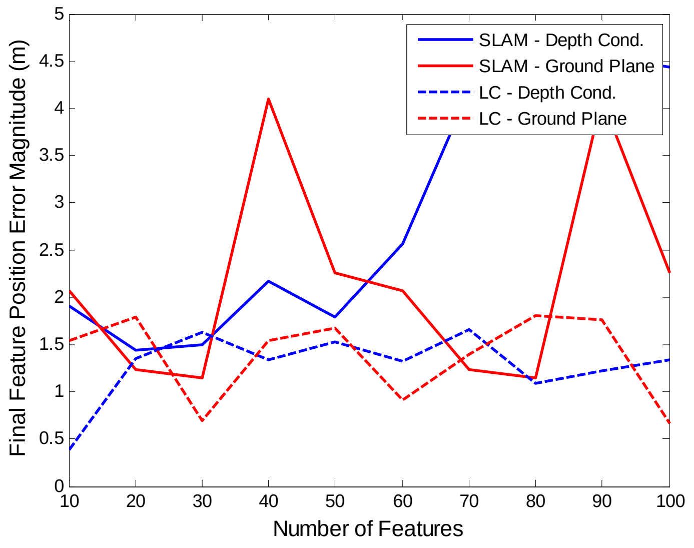


Figure 8-19: SLAM and LC Ground Plane and Depth Conditioning Final Feature Position Error Magnitude as a Function of Feature Density

Overall, the depth conditioning method's ability to accurately generate initial feature estimates and covariance matrices is better than the ground plane initialization. The ground plane initialization, however, represents a valid method if the local terrain approximates a relatively flat ground plane with high frequency terrain changes. Therefore, the depth conditioning method is used during experimental data analysis.

Nearly all results confirm that for this particular application a LC estimator provides the most robust results. This is especially true as the features are initialized based upon current aircraft states. In effect, biased aircraft estimates lead to biased feature initial position estimates. The correlation and relationships between the features and aircraft states then tend to increase this negative behavior. During the SLAM ground plane initialization analysis, an incorrect initial ground plane distance results in feature initializations that on average are biased, generally in the z direction. When most of the features are initialized below their actual positions, which happens when the distance to the nominal ground plane is overestimated, the aircraft's z position estimate starts to drift down toward the ground plane. Intuitively, this makes sense as camera measurement errors are minimized as the aircraft moves downward. However, this digression may lead to filter divergence, as future feature estimates are now even more biased. Therefore, for the remainder of this research a loosely coupled filter, as shown in Fig. 8-20, is used. Essentially, a basic GPS/INS UKF provides an estimate of the aircraft states from the fusion of accelerometer/gyro input measurements and GPS position/velocity correction measurements with the relationships defined by the dynamic (3.15) and observation (3.16) models. The Feature UKF is exclusively an observation model, and therefore is only executed when feature data from the camera image is obtained. The UT is used during the innovation covariance determination as the observation model for the features is dependent upon the aircraft state mean values and state covariance matrix. This estimator is robust in the sense that a divergent map will not adversely influence the aircraft states. However, this also means that any useful information possibly obtained from the features will not help correct the aircraft states.

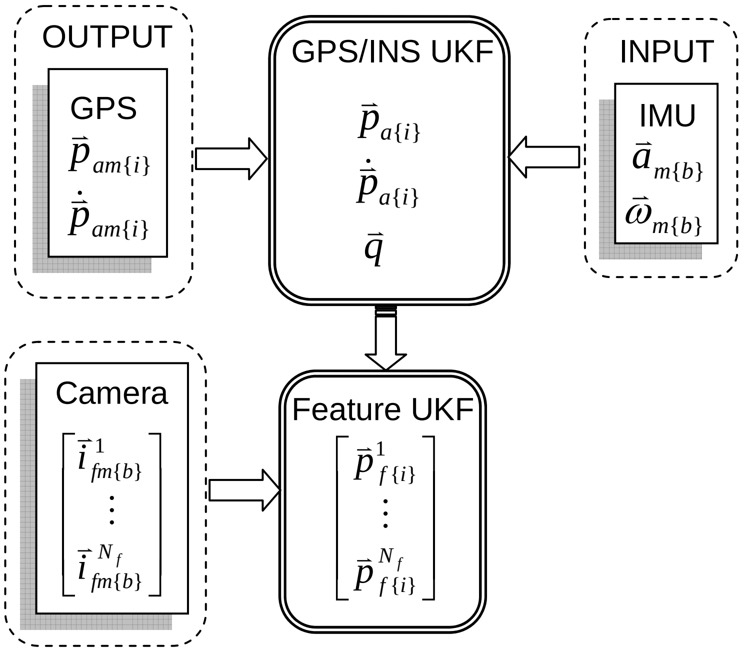

OUTPUT
GPS
$\vec{p}_{am\{i\}}$
$\dot{\vec{p}}_{am\{i\}}$
GPS/INS UKF
$\vec{p}_{a\{i\}}$
$\dot{\vec{p}}_{a\{i\}}$
$\vec{q}$
INPUT
IMU
$\vec{a}_{m\{b\}}$
$\vec{\omega}_{m\{b\}}$
Camera
$\begin{bmatrix} \vec{i}^{1}_{fm\{b\}} \\ \vdots \\ \vec{i}^{N_f}_{fm\{b\}} \end{bmatrix}$
Feature UKF
$\begin{bmatrix} \vec{p}^{1}_{f\{i\}} \\ \vdots \\ \vec{p}^{N_f}_{f\{i\}} \end{bmatrix}$

Figure 8-20: LC GPS/INS UKF and Feature UKF State Filters

Chapter 9: Experimental Results

While actual target tracking implementations need to execute in real-time, the data collected for this research are stored with the PC104 Stack data collection computer for post processing. Storing data for later processing is advantageous for several reasons. For one, programming glitches are easily debugged in a lab environment. Also, knowledge gained from hindsight is available. For instance, if a feature is only viewed in a few images then its importance to the filter is negligible as the feature's position will not be localized with only a few measurements. In fact, several images are generally required for feature initialization alone. Therefore, these features are never used by the filter or the filter's Feature Manager. This greatly reduces filter computation effort while still providing proof of concept analysis. Real-time applications, on the other hand, need to consider every feature because knowledge about the number of images the feature will be captured in is unavailable. Finally, a single data set can be analyzed and iteratively tested. This is beneficial since the process involved with collecting data is time consuming.

Data collection consists of a series of events. Before flight, the camera's shutter speed, aperture, focus, and mode settings are adjusted for the current lighting and flight conditions. The PC104 Stack is also initialized. The aircraft is then launched, and the telemetry is monitored with the ground station. The flight plan is chosen such that the aircraft traverses varying terrain, including rural roads and valleys common to the local area. The data collection starts upon reception of a command from the OI. The camera is triggered and the processor time is recorded by the autopilot at the instant new GPS data are available (approximately 4Hz). This allows the same observation equations to be used at each correction step of the filter. At each control iteration of the autopilot cycle, which executes at 20Hz, the current sensor data and CCT's GPS/INS solution are bussed across the CAN port to the data collection computer. The PC104 Stack also stores the image data and processor time associated with the image for post processing.

A few important modifications to the filter algorithms are necessary for experimental data analysis as compared to simulation. In the simulation, the pixel measurement locations for features and the target within each image are artificially generated. In the experimental data case, actual aerial images are used. A feature extraction algorithm is needed to identify the pixel coordinates of "keypoints" and to match these features in multiple images. Another change includes modifications to the feature management code. The manager is now responsible for feature initialization as discussed in the previous chapter. The process of initializing features has proven to be one of the greatest challenges concerning practical implementation of the methods in this research. Given the results of the simulations and limitations on the extent of the research, this chapter focuses on the LC GPS/INS UKF and Feature UKF. The tightly coupled, SLAM, filter and target tracking are not discussed here.

9.1 Feature Extractor

The UKF Feature Manager must know when a feature is observed in an image and also its corresponding pixel coordinates. A feature extraction algorithm generates this data from the raw images obtained during the data collection flight. One of the first steps involves the application of the scale invariant feature transform (SIFT) algorithm to each image. This algorithm returns the feature's location within the image and its descriptor. Since the extraction algorithm relies upon this feature detector and identifier application, knowledge about the functionality and limitations of SIFT is important.

9.1.1 Scale Invariant Feature Transform

A demo version of the SIFT algorithm based upon the work done by David Lowe is used in this thesis. This algorithm is designed to provide a robust feature detection and matching mechanism for overlapping images that is invariant to image scale, rotation, noise addition, and illumination. This is accomplished with a series of filtering stages designed to minimize cost while providing consistent and reliable results [20].

1. Scale-space Extrema Detection - In this stage, a Gaussian smoothing filter is applied across the image. The scale on the filter is varied and then applied to the same image. The difference between the two filtered images, known as the difference of Gaussian, is

taken. This method provides a means to detect points of interest, also known as keypoints.

2. Keypoint Localization – Once a keypoint is determined, a quadratic function is used to approximate the location of the point. This allows the location to take values that are fractions of a pixel, which enhances keypoint detection stability and matching reliability. The local gradients are then analyzed to eliminate edge points. These are places where the gradient is very prominent in one direction while very poor in another direction.
3. Orientation Assignment – In order to account for image rotation, each keypoint is described relative to its distinctive orientation direction. The orientation is determined by calculating gradient changes in the Gaussian filtered image about the keypoint. The scale chosen to filter the image for this calculation is the one that provided the extrema detection point from the first stage.
4. Keypoint Descriptor – Each keypoint is assigned a descriptor, or personal identity. Numerically, the descriptor is a 128 element normalized vector that describes orientation gradients about the keypoint. These gradients are calculated relative to the orientation assignment generated in the previous stage. The vector is normalized to reduce the effects of illumination.

While the SIFT algorithm determines keypoint pixel locations in images and returns the corresponding descriptor for each feature, a metric is needed to determine reliable feature matches in multiple overlapping images. Keypoint matching is accomplished by taking the vector dot product of every keypoint descriptor in an image with every keypoint descriptor in another overlapping image. Since the descriptors are normalized, taking the arccosine of the dot product yields what will be loosely called the “angle” between the two vectors. The angles, formed from a single descriptor in the first image and all the descriptors in the second overlapping image, are then ordered based upon size. The smallest angle denotes the best match. However, the match must be distinctive. This means that the next closest match should be a significantly larger angle. Otherwise, obtaining a false match is probable. The matching function used in this feature extractor requires the smallest angle to be at least 40% smaller than the next smallest angle. If this percentage is smaller, many false matches occur while increasing the percentage disregards many correct matches.

An example of SIFT's ability to extract and match features across overlapping images is demonstrated in Fig. 9-1 and Fig. 9-2. The images, plotted side by side in both figures, were captured during one of the ECat's data collection flights over rural terrain. The SIFT algorithm extracted 662 keypoints from the leftmost image and 536 keypoints from the rightmost image as marked by the red circles in Fig. 9-1. As expected, these points are located along areas of sharp contrast, such as the rock outcropping and the gully. In other less distinctive areas where the ground terrain is not as rugged and the plant growth is uniform, very few features are found.

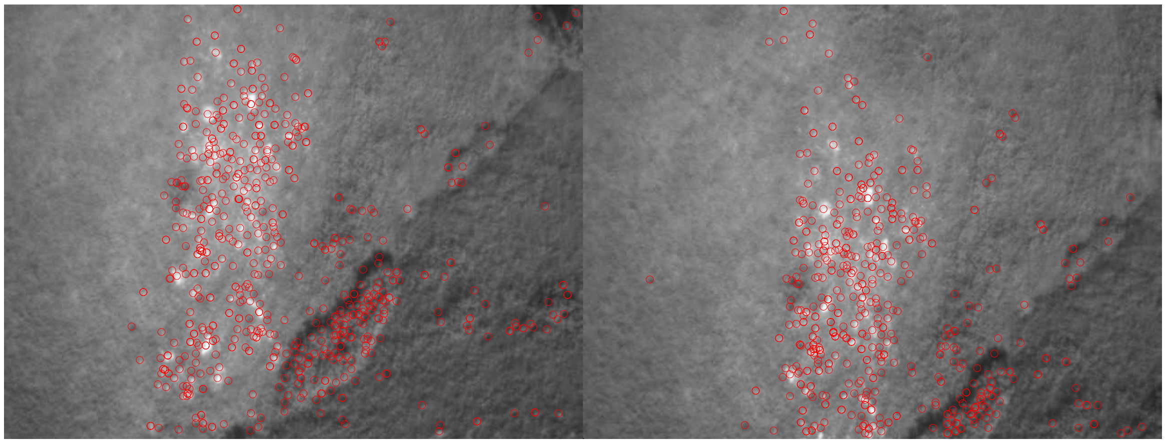


Figure 9-1: SIFT Features

The following figure displays the 168 feature matches connected by lines, as determined from the SIFT descriptor matching function. In this case, the descriptors on the leftmost image are compared to all of the descriptors in the rightmost image. Since the difference between the two images is mostly translation, the lines between all correct matches should be nearly parallel. One can observe through qualitative assessment of the figure that the matching function provides a good method of determining correlation between keypoints.

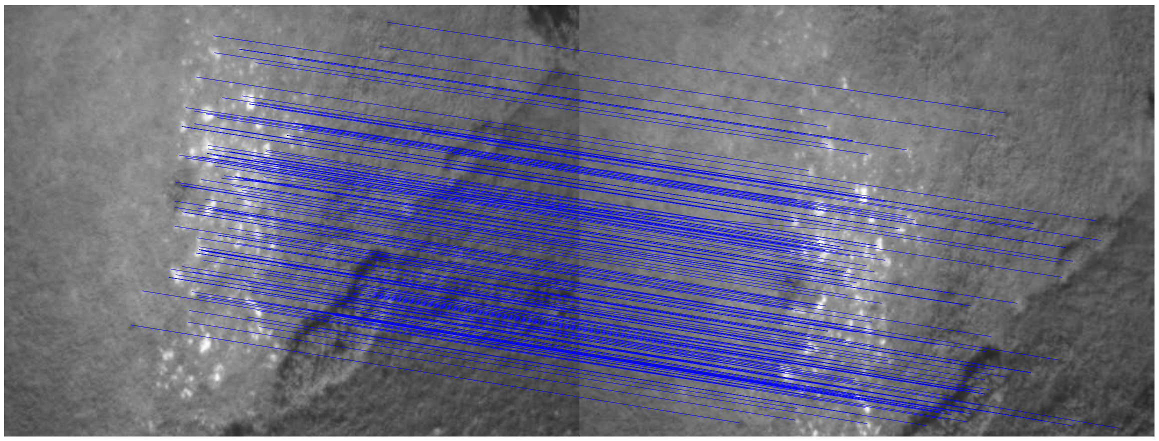


Figure 9-2: Feature Matches

9.1.2 Extraction Algorithm

The extraction algorithm is responsible for performing all the tasks from importing the images to supplying a robust list of feature information to the Feature Manager. This algorithm is built for identifying keypoints with SIFT and uses a series of logical comparisons to determine valid matches. Since the SIFT keypoint matching algorithm is susceptible to false matches, a three image forward filter provides a series of tests aimed at limiting the quantity of these anomalies. Ultimately, the Feature Extractor must provide the Feature Manager with the output listed in Table 9-1. The first variable in the table is the global number assignment. Since SIFT only provides a vector of keypoints for each image, a global number assignment is necessary for reference and organization. The start and size variables are used to denote which consecutive images the feature is located within. This implies that if a feature goes out of view and later comes back into view, it will be given a new feature number. This has a major advantage. The keypoint descriptor for each image does not need to be stored in a database and checked indefinitely with new image features. This decreases computational time required during extraction, simplifies the complexity of the code, and prevents an increased likelihood of false matches. The last variable refers to the actual feature pixel coordinate locations needed by the Feature Manager initialization code and the correction steps of the Feature UKF. This method would need slight modification for real-time implementation, but does not violate the principle of using only past and current knowledge within the state estimator.

Table 9-1: Feature Manager Required Variables and Definitions

Variables	**Definition**
Global Number	The unique value assigned to each feature
Start	Image number when feature is first observed
Size	Number of consecutive images with feature
Pixel Locations	x and y pixel locations of feature within each image

9.1.2.1 High Level Block Diagram

The block diagram in Fig. 9-3 provides a high level perspective of the feature extraction algorithm. After images are imported, the three image forward filter is initialized. This involves analyzing the first three images such that the necessary data structures required by the main filter are readied. The three image forward filter, which is the most complex section of the algorithm, then starts to process each image.

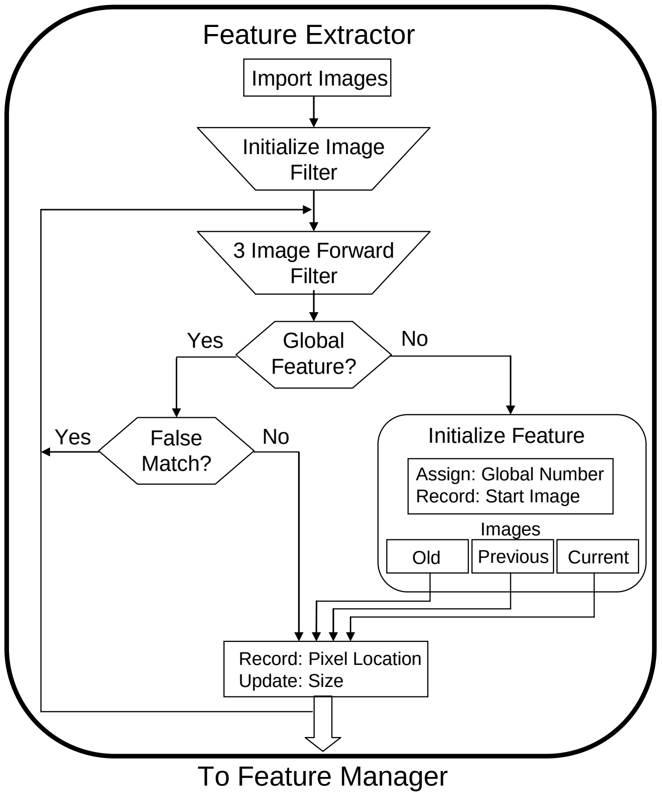


Figure 9-3: Feature Extractor Block Diagram

The following example demonstrates the method by which the three image forward filter determines valid matches. The general concepts involve comparing keypoints across three images, such as those shown in Fig 9-4. The images are named Current, Previous, and Old. The Current image represents the image most recently captured. The Previous image was recorded in the previous frame while the Old image is two frames old.

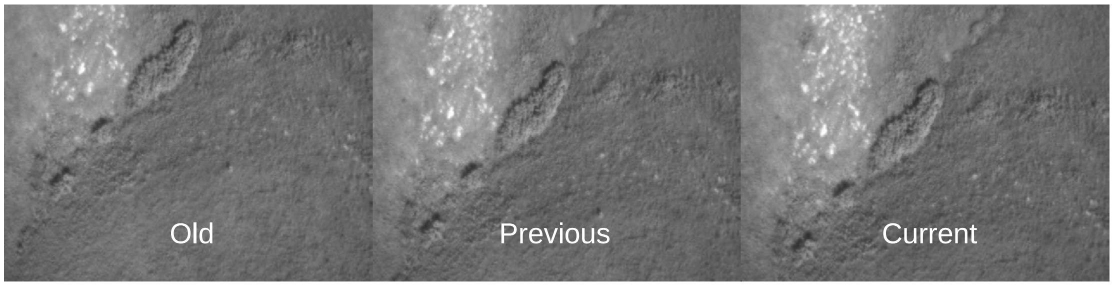


Figure 9-4: Three Image Example

The descriptors for all keypoints in the three images are determined with the SIFT algorithm. The first matching function check occurs between the Old and Previous image keypoint descriptors. The matching function returns a vector, $\vec{m}_{OP}$, which contains all of the keypoint matches between the two images. This vector has a length equal to the number of keypoints in the Old image where each row corresponds to an individual keypoint in the Old image. If a match has occurred, the keypoint number in the Previous image is displayed in the vector, otherwise the element is null. Equation (9.1) shows the first few elements of a matching vector for the Old and Previous photos.

$$\vec{m}_{OP} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 110 \\ \vdots \end{bmatrix} \tag{9.1}$$

Since the first element is zero in $\vec{m}_{OP}$, no keypoint in the Previous image matches the first keypoint in the Old image. Element 2, however, is nonzero. Therefore, the second keypoint in the Old image matches the 4 element in the Previous image. This is illustrated in Fig. 9-5 where the keypoints are numbered.

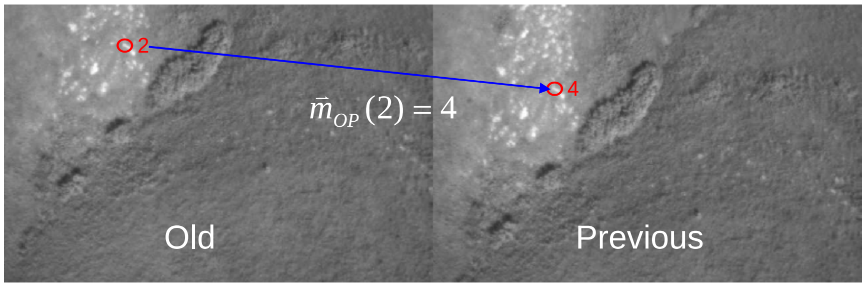


Figure 9-5: Match between Keypoint in Old and Previous Images

The next check involves determining if the keypoint, found in both the Old and Previous images, is also located in the Current photo. Therefore, the fourth element of the Previous and Current image match vector, $\vec{m}_{PC}$, is checked where each row in this vector corresponds to a keypoint in the Previous image.

$$\vec{m}_{PC} = \begin{bmatrix} 18 \\ 0 \\ 0 \\ 3 \\ \vdots \end{bmatrix} \tag{9.2}$$

The fourth keypoint in the Previous image does match a keypoint in the Current image, specifically the third keypoint. Once again, this is shown pictorially in Fig. 9-6.

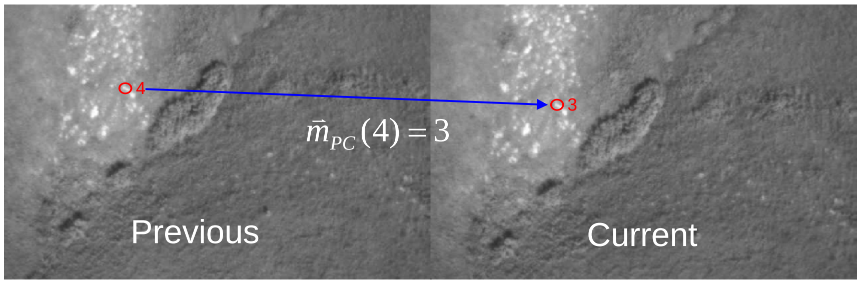


Figure 9-6: Match between Keypoint in Previous and Current Images

One final check exists. The original keypoint from the Old image is matched with the keypoints in the Current image, yielding the last matching vector, $\vec{m}_{OC}$. According to $\vec{m}_{OC}$, the second keypoint in the Old image matches the third keypoint in the Current image. This is the same keypoint in the Current image determined by the previous two matching checks.

$$\vec{m}_{OC} = \begin{bmatrix} 0 \\ 3 \\ 10 \\ 21 \\ \vdots \end{bmatrix} \tag{9.3}$$

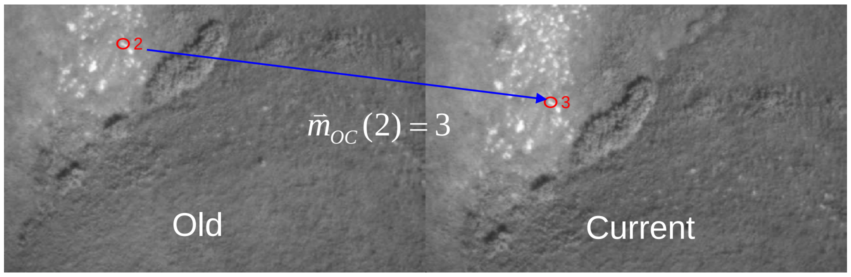


Figure 9-7: Match between Keypoint in Old and Current Images

All of the required matching relationships for the three image forward filter are summarized in Fig. 9-8. By requiring all of the logical matching checks, shown as vectors in the figure, the intention is to reduce the likelihood that a falsely matched keypoint passes all the tests.

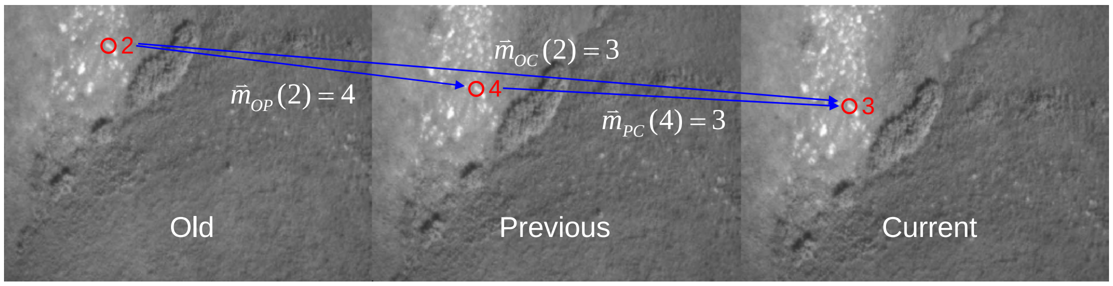


Figure 9-8: Match Relationships between a Keypoint in All Three Images

Referring to Fig 9-3, once a keypoint has passed the requirements of the three image forward filter, it is checked against the current global database to determine if it is a new feature or a previously initialized global feature. When the feature is not recognized within the database, it goes through a quick initialization procedure. This consists of giving the feature a unique global number and recording the initial image number (Old image number). The pixel coordinates of the feature located in the Old, Previous, and Current images are also recorded, and the feature size is incremented by three. If the keypoint already contains global status and passes an additional logical test, then the Current image pixel coordinates for the feature are recorded and its size is updated by one. The additional logical check helps identify a very rare problem, which is illustrated below in Fig. 9-9.

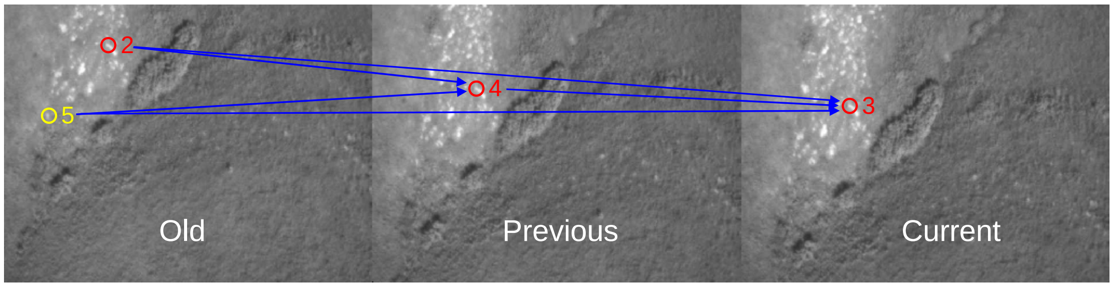


Figure 9-9: Rare Keypoint Matching Error

In this case, the fifth keypoint in the Old image passes the three image forward filter tests. Keypoint two also passes this test and represents the true match. This problem usually originates when the keypoint descriptor for keypoints two and five are very similar. One method to eliminate this problem includes operating the matching function in both a forward and reverse manner. When the matching function tries to find a distinctive match for keypoint descriptor four in the Previous image with all of the keypoint descriptors in the Old image, two points will be of interest (two and five). The matching function, however, is distinctive. Therefore, no match is returned in this case since descriptors for keypoints two and five in the Old image generate similar matching results with the fourth keypoint descriptor in the Previous image. Since this problem is rare and additional matching and logical comparison increases processing time, another method is utilized in this extractor for identifying the problem. This method looks at the size and starting image of the global feature. If this sum of the starting image number and current feature size is one greater than the current image number, the feature has already been updated, and a possible false match has occurred. Even if a false match occurs in this part of the algorithm, the global feature is updated correctly as only the pixel coordinates for the Current image global filter are needed. This is not guaranteed if both keypoints two and five of the Old image have never been initialized. In this case, whichever feature passes the three image forward filter test first has its Old image pixel coordinates stored.

The extractor algorithm continues to iterate until all images are processed. In the three image forward filter, the Previous image becomes the Old image, the Current image becomes the Previous image, and the newest picture is designated as the Current image. Once all the data are generated, the features are weighted based upon their number of observations. They are also weighted based upon the number of other features viewed in an image. For instance, a feature that is viewed in an image with very few other features is given a higher weight than another feature that is viewed in images with several features. By doing this, a computationally efficient set of features are used within the Feature UKF. The results of the extractor still apply if all of feature data are provided to the Feature Manager, and therefore this feature weighting and sorting step is not shown in the block diagram. Finally, the pertinent features are stored in the appropriate format required by the Feature Manager.

9.2 Feature Manager

The Feature Manager is a function within the Feature UKF algorithm that is called when correction data are available. This occurs before the actual prediction step, as some features are no longer observed and other features are ready to be estimated. The major tasks demanded of the Feature Manager are given here.

- Track feature status.
- Update the mean vector and covariance matrices.
- Initialize feature mean and covariance.

9.2.1 Status Description

Each feature is given a status flag. The values that this variable accepts along with short description of its meaning are located in Table 9-2.

Table 9-2: Status Flag Values and Descriptions

Value	Description
0	Feature has not been observed
1	Feature observed at least once, but not initialized
2	Feature observed, initialized, and being estimated
3	Feature estimated and no longer observed

Every feature imported from the Feature Extractor starts with a “0” status. Once a feature is first observed in an image, the status variable is updated to “1”. Also at this time, the current position and orientation of the aircraft and associated covariance terms are stored along with the current pixel location measurements. Before the next status level is reached, the feature must be initialized through one of three techniques discussed in the previous chapter. The number of poses required for the initialization is variable depending upon the method used. Once the initialization procedure has deemed the feature acceptable for initialization, the feature obtains “2” status. This concludes with an augmentation of the Feature UKF mean vector and covariance matrix. Finally, once a feature is no longer observed, pertinent mean and covariance information is extracted from the main filter and the final status level is awarded.

9.2.2 Updating Mean Estimate and Covariance Matrix

The state estimate vector and state covariance matrix are allowed to dynamically change based upon current feature status. Once a feature's initial mean and 3x3 covariance matrix have been properly identified, the mean is augmented to the end of the current state vector, and the covariance matrix becomes the lowest block diagonal element of the augmented error covariance matrix. As outline in section 6.2, the new error covariance matrix is positive semidefinite given the original state covariance matrix and 3x3 initial feature covariance matrix are also positive semidefinite.

9.3 Results

The data discussed in this section were collected during a flight on the 8th of November 2007. The flight covered a rural section of local prairie with heavily undulating terrain. The UAV was manually piloted for the duration of the mission where the pilot tried to maintain a fairly constant altitude while navigating the aircraft over the terrain in a loop pattern oriented from the Southwest to the Northeast. The experimental data were stored for nearly three minutes of this flight, resulting in 614 aerial images captured during GPS updates and approximately five times as many IMU data packets. The features were extracted using the Feature Extraction algorithm discussed previously. The 100 most viewed features were kept for use in the LC GPS/INS UKF and Feature UKF estimators.

The basic GPS/INS UKF uses aircraft dynamic and observation equations in conjunction with the accelerometer/gyro and GPS position/velocity measurements to obtain aircraft state estimates. The filter also requires the input and output measurement noise statistics associated with the aircraft equations as given in Table 7-1. Unfortunately, the true aircraft states are unavailable for error calculations. However, CCT's GPS/INS is available for comparison. Their solution incorporates the use of the industry standard EKF for state estimation, and any following reference to the EKF refers to CCT's state estimates.

Several plots comparing these two aircraft navigation solutions are located in Fig. 9-10 through Fig. 9-14. The position plots compare the GPS sensor reading with both the UKF and EKF solutions where all locations are given relative to the base station. In general, the East and North distance position estimates and measurements are nearly identical. This is expected as the precision dilution of precision, or satellite constellation, typically allows accurate latitude and longitude GPS estimates, which are weighted more significantly within the navigation solution. However, the accuracy of the GPS altitude estimates is less certain. As illustrated in Fig. 9-10, the UKF and EKF altitude estimates vary more than the East and North position estimates. The GPS measurement covariance statistics used within the GPS/INS UKF are determined from the high frequency GPS estimate noise. CCT's EKF estimator uses a larger more conservative covariance term, which captures the low frequency drift of GPS altitude measurements. Thus, the GPS measurements carry more merit within the UKF estimator than the EKF estimator. This is the reason the UKF solution conforms more closely to the GPS position measurements, especially with respect to altitude. Also shown are the three Euler angle plots, which include the roll, pitch, and yaw of the aircraft. In order to avoid clutter, only the first 100 seconds of the flight are shown. After approximately ten seconds, all UKF angle estimates mimic the EKF angle estimates. No other observation measurements are available for this comparison as in the position plots.

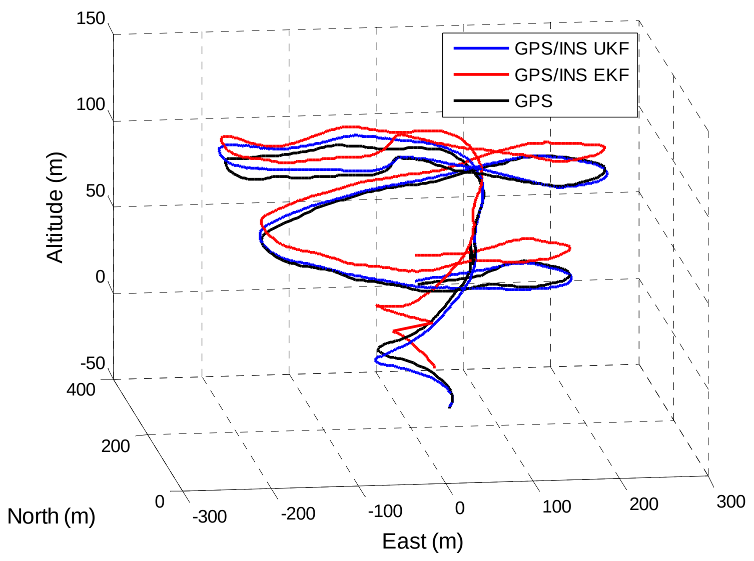


Figure 9-10: 3D GPS and Position Estimates

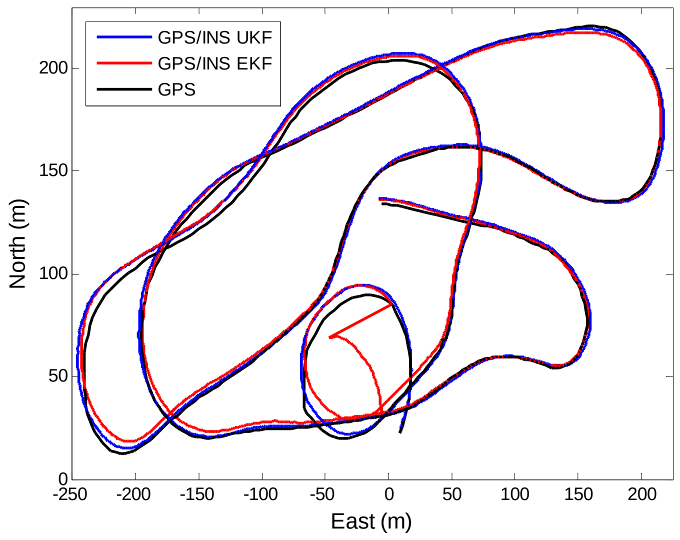


Figure 9-11: 2D GPS and Position Estimates Overhead View

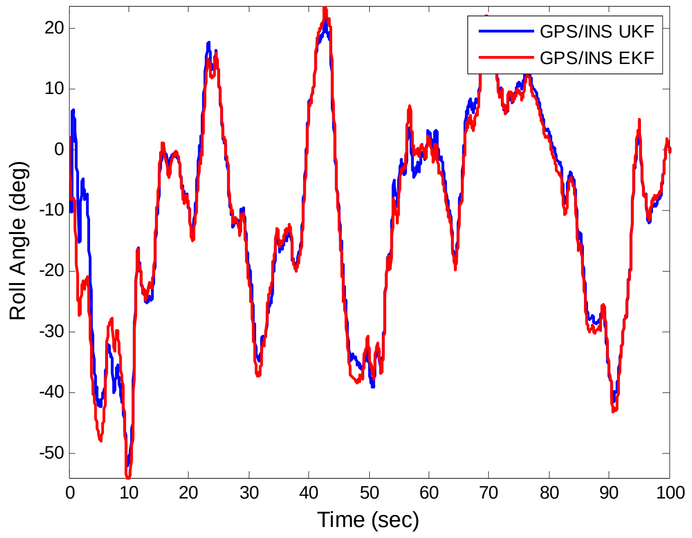


Figure 9-12: Roll Angle Estimates

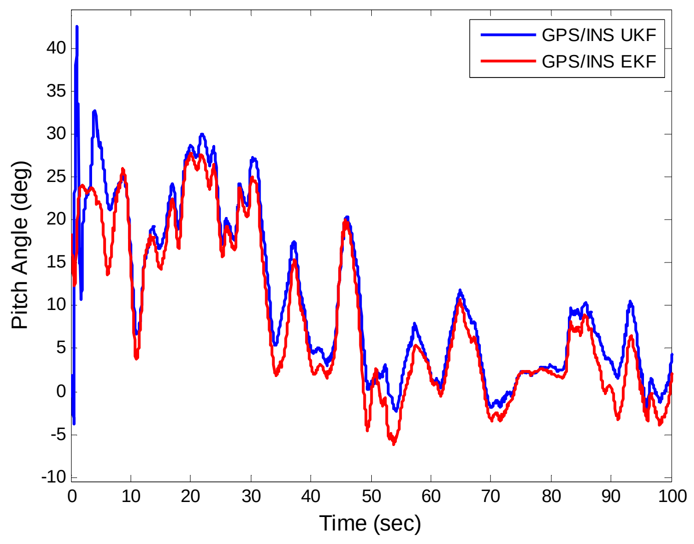


Figure 9-13: Pitch Angle Estimates

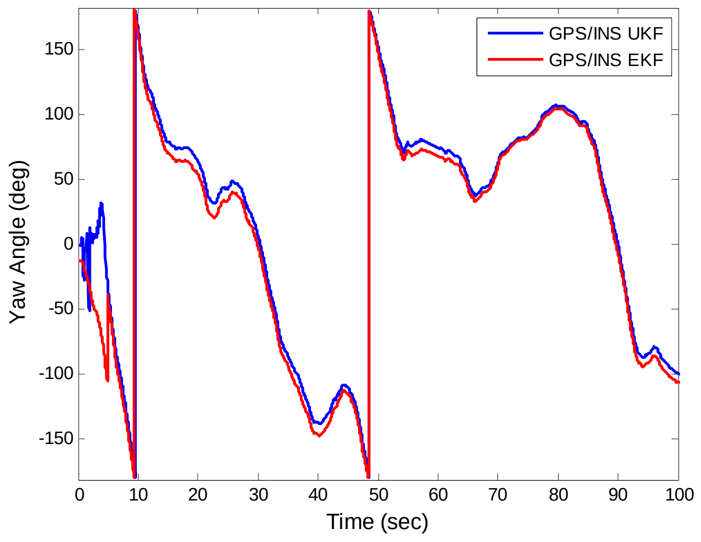


Figure 9-14: Yaw Angle Estimates

The Feature UKF relies upon the aircraft position and orientation estimates and image observations to construct a terrain map of the local environment. The depth conditioning method is employed for feature initialization. Two hundred particles are used with a base lambda of 150m and a spread of 200m. This base lambda is determined from the mean altitude and FOV of the camera lens combo. The initial mean position estimates for the features are stored when the feature’s states are augmented to the filter estimate. Upon final observation, each feature is removed from the estimator, and its final mean value is stored for later evaluation. Sorting algorithms allow extraction of features that meet a certain criteria, such as a specific number of corrections.

With the resources available for this research, a quantitative measure of the feature estimator performance was not possible. For this reason two general comparison techniques are performed. The first comparison consists of determining the validity of the feature locations as geo-referenced to a Google Earth satellite image. The feature locations within the geo-referenced images are then compared qualitatively with actual high resolution aerial photography images obtained during the data collection flight. The other comparison considers only altitude correctness. This comparison uses topographic data collected with Light Detection and Ranging (LIDAR) measurements available for the area.

A geo-referenced satellite image from Google Earth fills the background of Fig 9-15 and Fig. 9-16. This fairly high resolution image displays distinct landmarks in the vicinity of the UAV’s flight path, denoted by the blue line. All North and East distances are given relative to the location of the base station. Because the ground area encompassed by the image is relatively small compared to the Earth’s surface, the linear axes capture image scales well within the accuracy of any discussions performed here. The linear scale is found using a constant latitude and longitude conversion factor based on the base station location. The blue circles in Fig. 9-15 correspond to the initial 2D feature locations for all features that pass through the correction step of the filter at least 10 times. The red circles in Fig. 9-16 denote the final mean estimate for these same features. The global feature numbers for a few features are plotted slightly due East of their positions. For subsequent analysis, only these numbered points are considered due to their ease of identification within the photos.

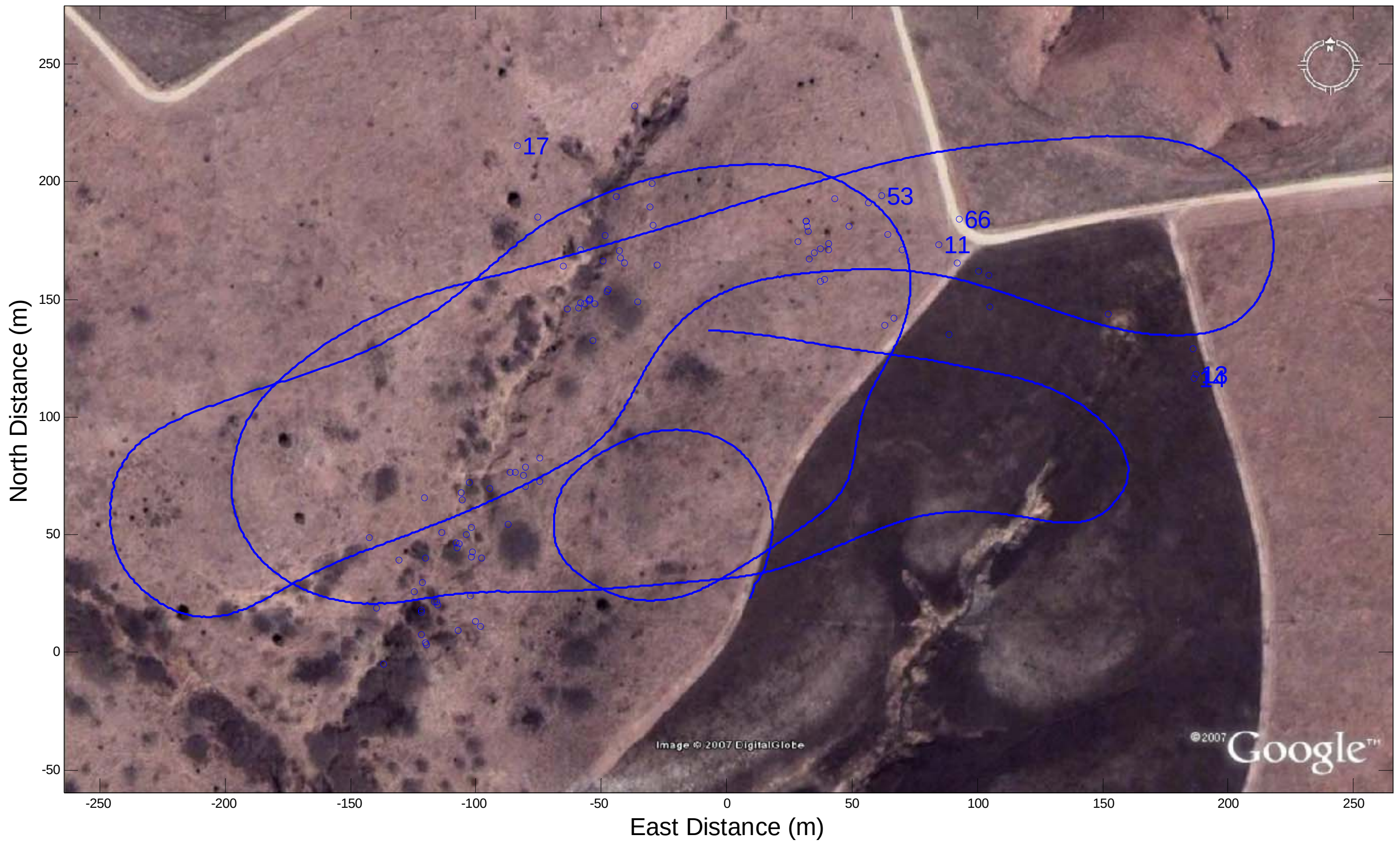


Figure 9-15: Geo-referenced Estimator Initial North and East Feature Locations

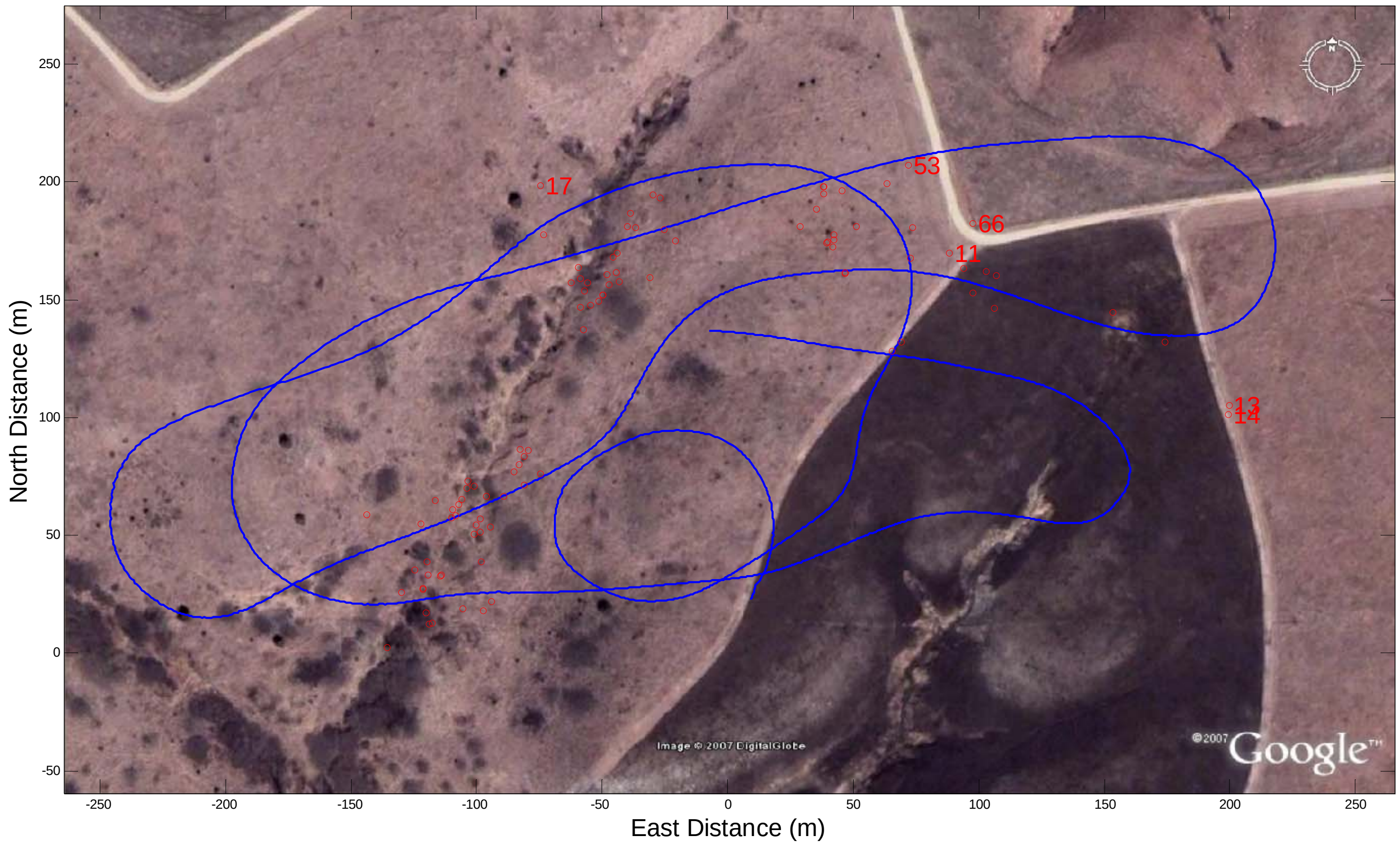


Figure 9-16: Geo-referenced Final North and East Feature Locations

Feature 17 is displayed in the high resolution aerial image in Fig. 9-17. The feature's location in this image is given strictly by the value returned by the SIFT algorithm. Two distinct landmark references are discernable from within the image: the cedar tree to the lower left of the feature and the start of a washout to the upper right of the feature. Both the tree and the washout are located in the Google Earth image. The initial feature location, obtained from Fig. 9-15, is obviously incorrect. It is located too far to the North and West of the actual location. The final location is more believable as shown in Fig. 9-16, where the feature is located somewhere in between the two distinct landmarks.

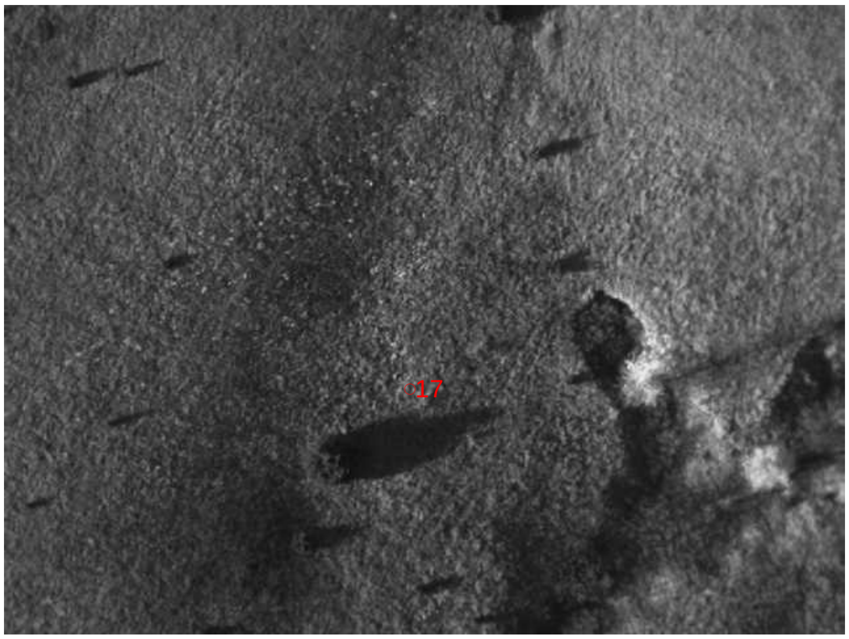


Figure 9-17: Aerial Image of Feature 17

Features 11, 53, and 66 are shown in another aerial image, Fig. 9-18, where the top of the image corresponds to generally a Southerly direction. Both the initial feature estimates for 11 and 66, displayed in Fig. 9-15, appear located too far Northwest of their correct locations. In fact, the initial position of feature 66 is located in the center of the road. The initial position of feature 53 is also located very far from its apparent location. In general, the final estimated feature locations as illustrated in Fig. 9-16 are a better match to Fig. 9-18 than the initial position estimates shown in Fig. 9-15.

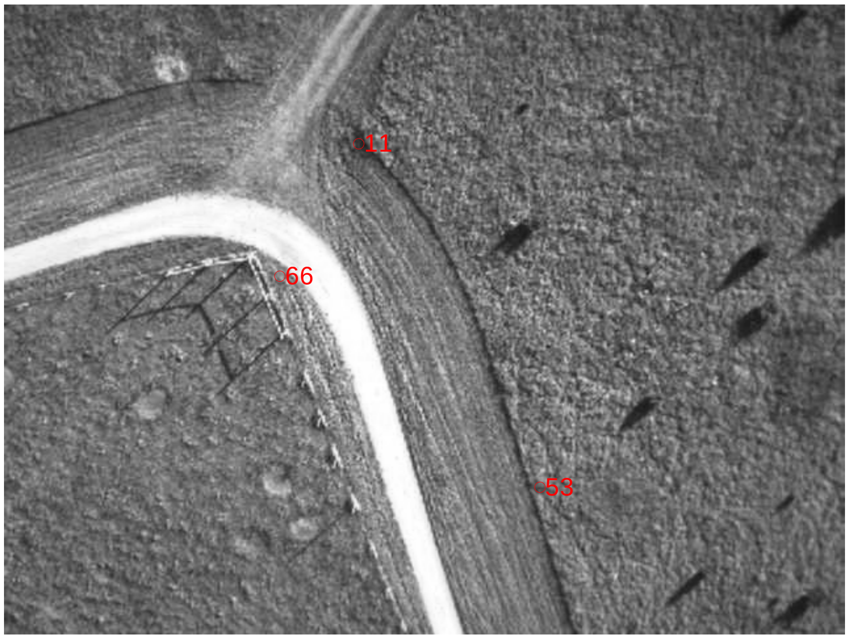


Figure 9-18: Aerial Image of Features 11, 53, and 66

This same trend is noticeable when analyzing features 13 and 14, as observed in Fig. 9-19. This picture is also oriented such that the top of the image is further South than the bottom. Initially, the features are not even located on this road, as evident in Fig. 9-15. However, the final locations as referenced in Fig. 9.16 are more highly correlated with Fig. 9.19.

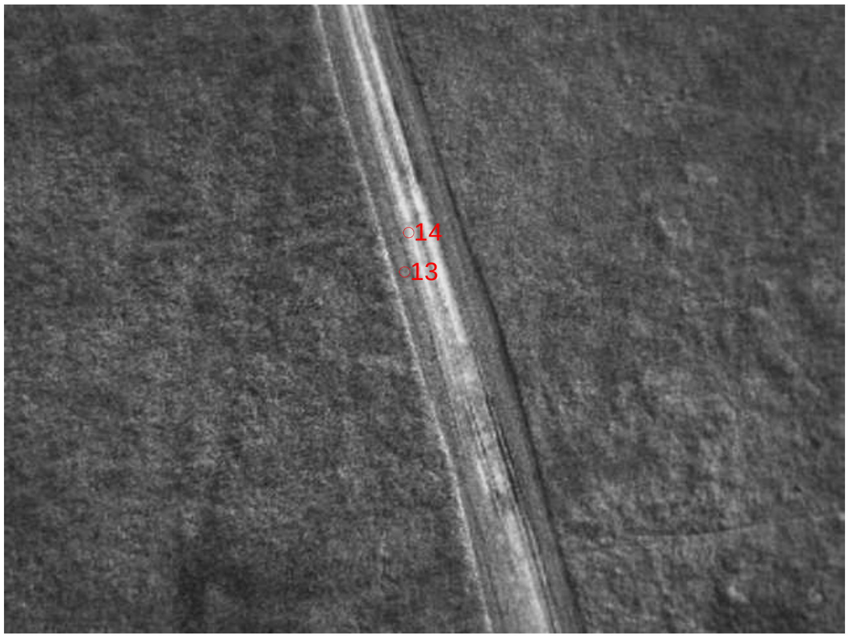


Figure 9-19: Aerial Image of Features 13 and 14

Overall, qualitative analysis of several other features in the same manner as presented here confirms that the Feature UKF does estimate feature location in the North and East directions to within a few meters of that predicted by the Google Earth images. The GPS estimates of latitude and longitude provide fairly precise knowledge of the aircraft's location relative to the base station. Even with a fairly small FOV camera, this allows the general 2D position of the feature to be determined. Depth reconstruction, however, is more difficult to determine.

The 3D perspectives of Fig. 9-15 and Fig. 9-16 are given by Fig. 9-20 and Fig.9-22, respectively. However, depth is still difficult to acquire from these figures. For this reason, 2D plots of the feature altitude and East distance relative to the base station are shown for the initial, Fig. 9-21, and final, Fig. 9-23, feature positions. As is readily determinable from the figures, the initial spread of altitude feature locations is very sizeable. Some of the features are initialized 40m above the base station altitude. Knowing that the base station is located along the top of a ridge, these features are obviously initialized far above their actual altitude. In fact, the variation in elevation of all of the roads in the high resolution images is well within a couple of meters, and the base station antenna is approximately two meters above the road. Correspondingly all the features discussed, except 17, should be very near zero meters in altitude. Although the final altitudes for the features discussed show more variation than this, they are obviously greatly improved. Furthermore, the final feature altitude locations do not exhibit the same magnitude of spread that is systematic in the initial altitude plots. The rough shape of the gully is even crudely approximated by this graph. The change in the initial and final altitude estimates for feature 53 is also of interest. The initial altitude estimate is vastly different from those of features 11 and 66. However, knowledge of the local terrain and consultation of Fig. 9-18 confirm that features 11, 53, and 66 should have nearly constant elevation values. The final feature altitude estimates agree with this statement as shown in Fig. 9-23.

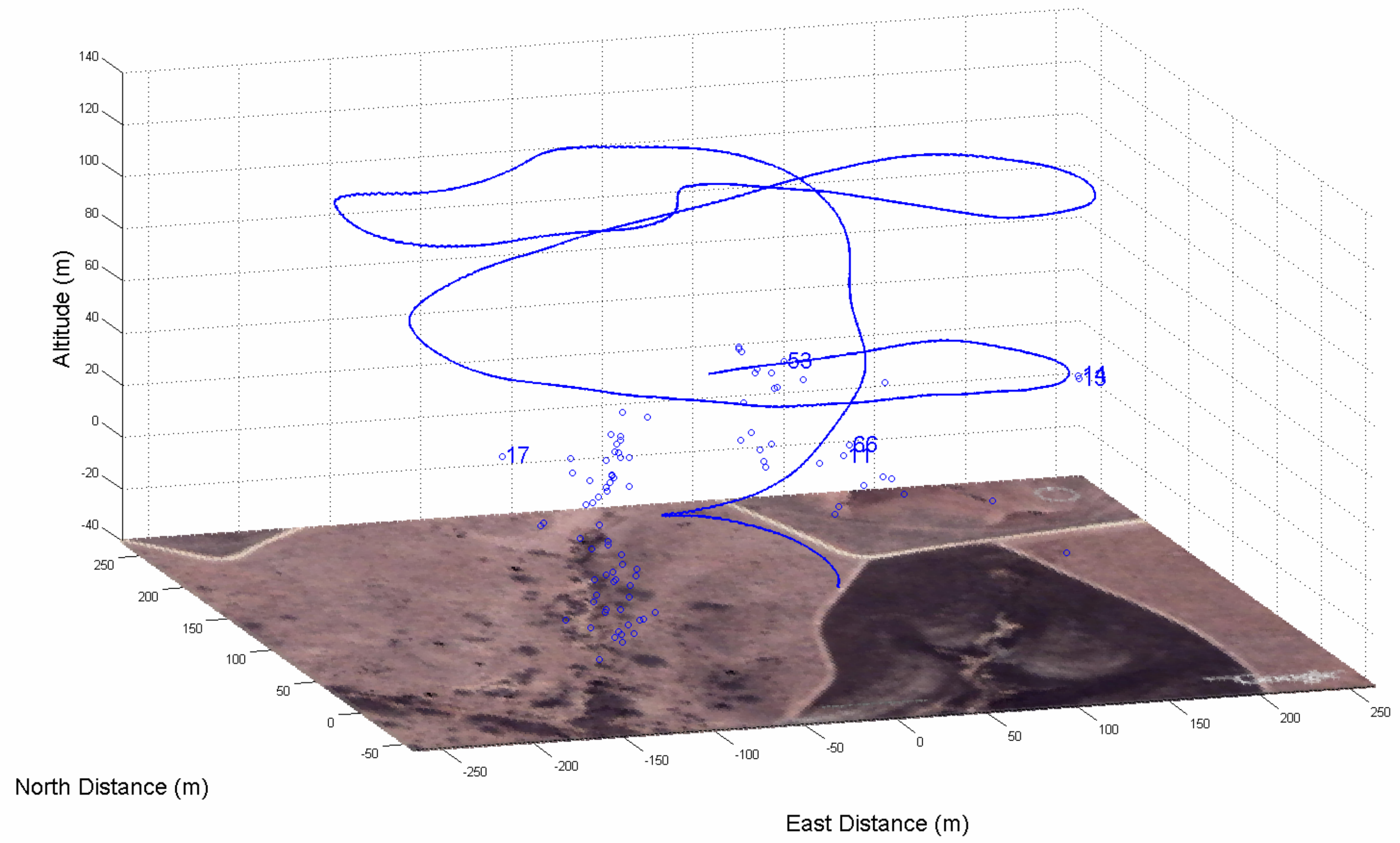


Figure 9-20: Geo-referenced Initial 3D Feature Locations

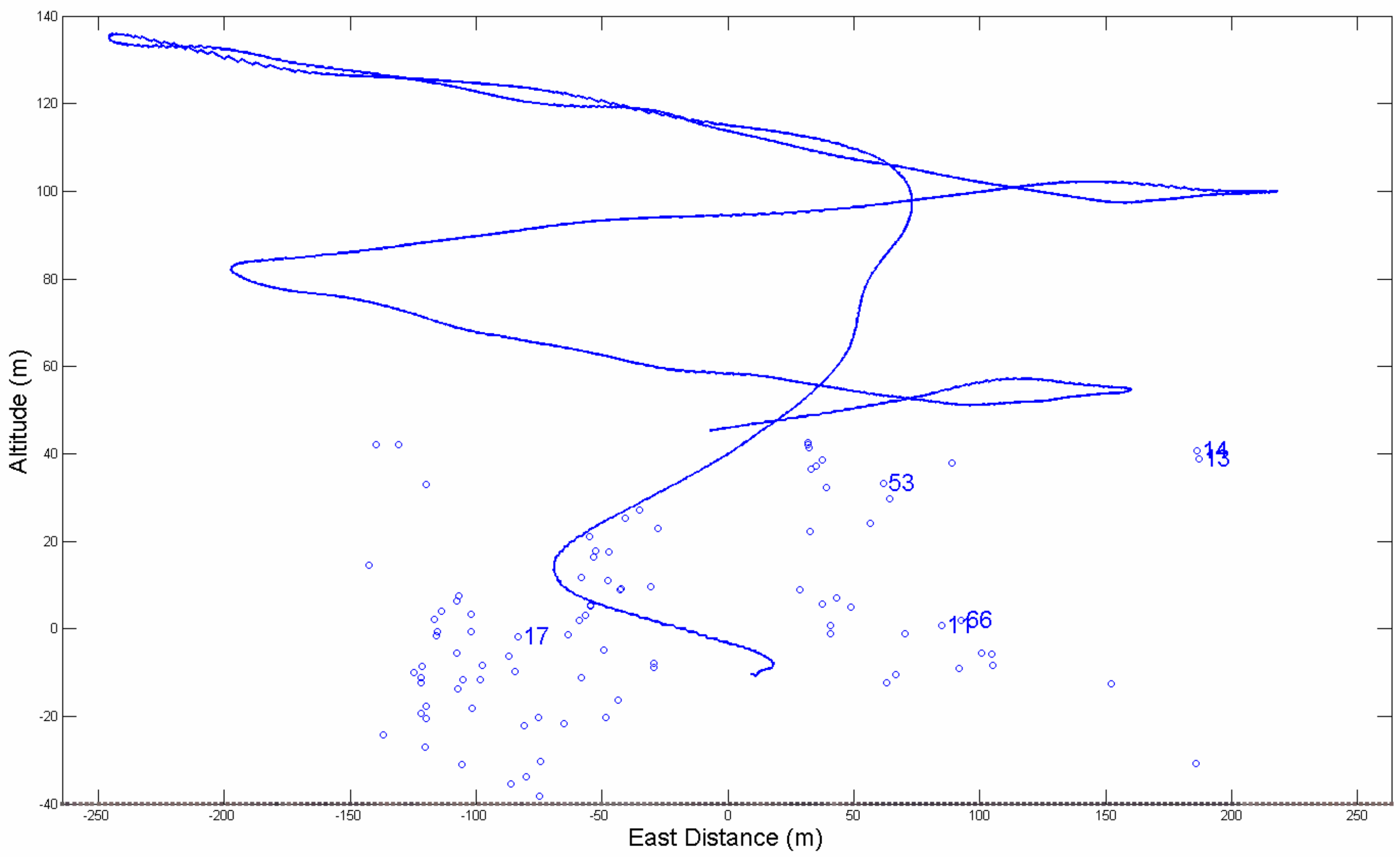


Figure 9-21: Geo-referenced Initial East and Altitude Feature Locations

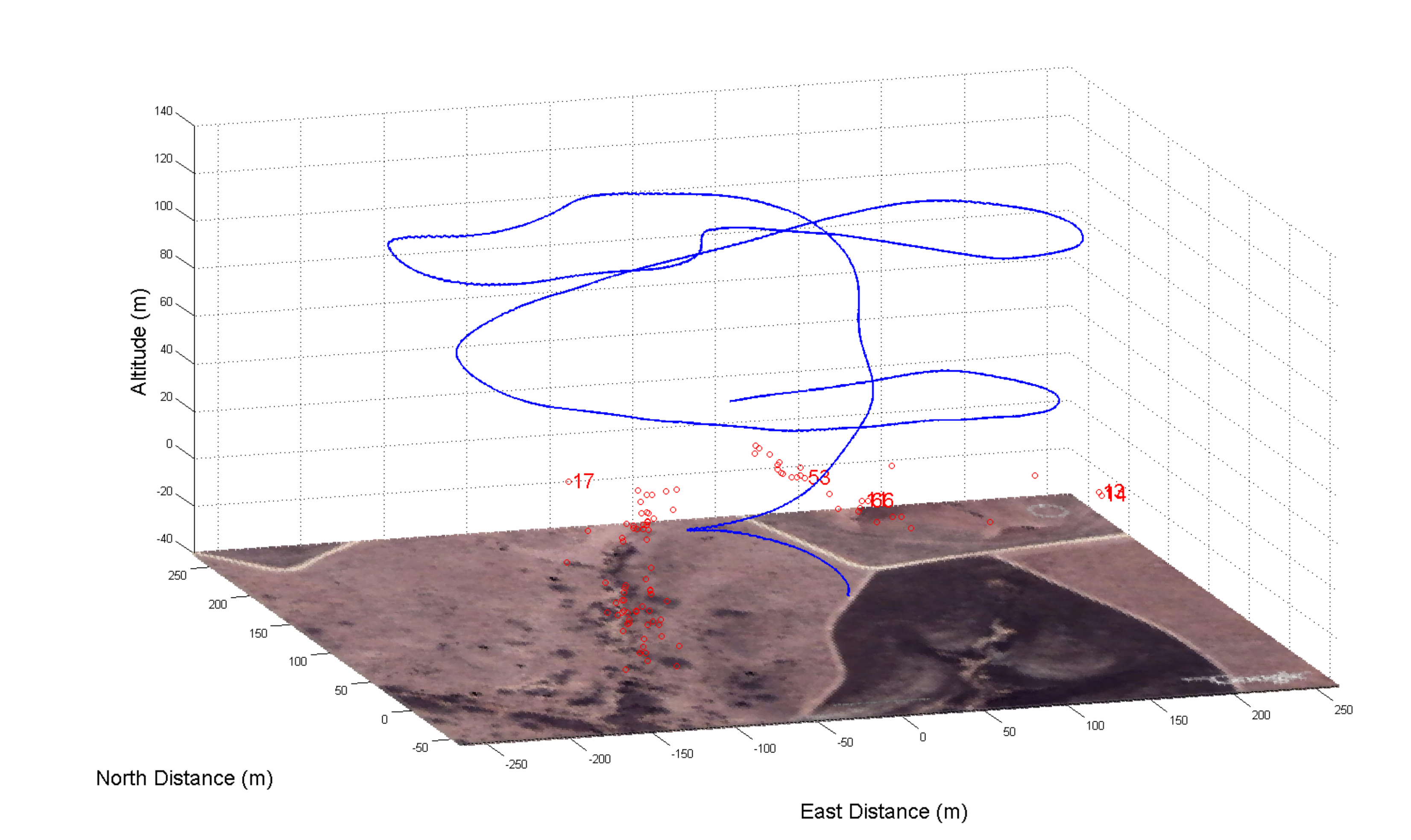


Figure 9-22: Geo-referenced Final 3D Feature Locations

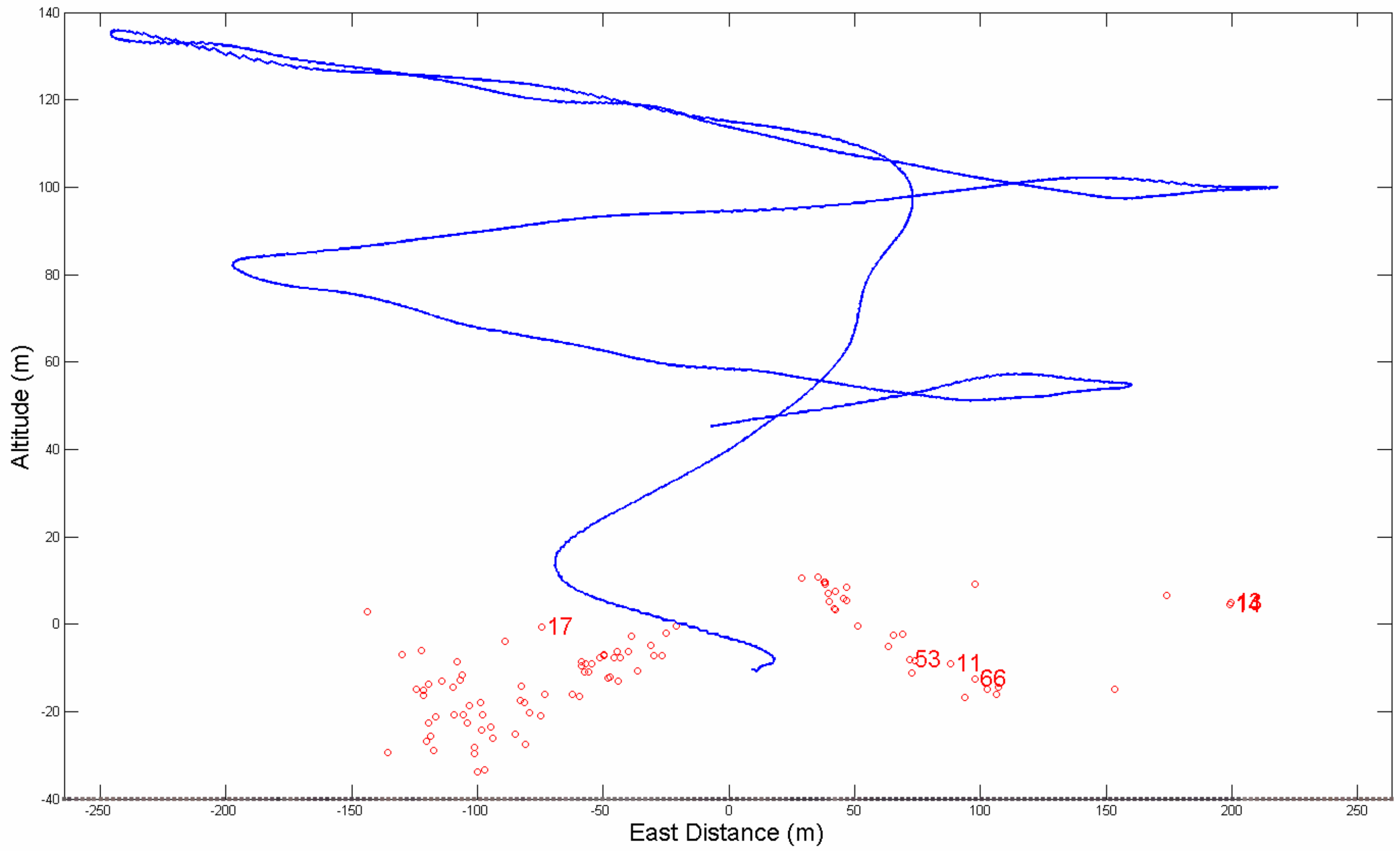


Figure 9-23: Geo-referenced Final East and Altitude Feature Locations

While the final feature locations appear more accurate than the initial locations, they still contain significant altitude errors. A somewhat more quantitative analysis can be performed using comparisons with LIDAR terrain data. The terrain data consists of several points, described in terms of the universal transverse mercator (UTM) coordinate system. The final feature latitude and longitude coordinates, which are believed to contain relatively small errors, are readily converted into the UTM coordinate system. The UTM feature locations and LIDAR terrain data points are plotted in Fig. 9-24 and Fig. 9-25. Both figures are rendered from the same perspective as given in Fig. 9-22 and Fig. 9-23, respectively. In general, the spread of the feature altitude values is very large. This reflects the higher uncertainty associated with the GPS altitude estimates and the inability of the image measurements to provide scale.

An average altitude error is determined for each feature as follows. The four closest surrounding LIDAR data points to each feature based solely upon the North and East UTM coordinates are located. The average altitude of these four closest LIDAR data points are then calculated and used as the “true” altitude for that feature. The average absolute value of the error between the averaged LIDAR altitude and the estimated feature altitude is shown in Fig. 9-26. This error is calculated for several different feature update values. The feature update values refer to the minimum number of times a feature is updated (i.e. passed through the correction step of the Feature UKF). If the feature update number is less than a certain value, then its error is not considered. The goal of this test is to determine how strong of a correlation exists between altitude error and the number of updates. As the number of corrections increases, the average altitude error per feature decreases as expected. Unfortunately, the number of features corrected decreases as the correction number increases. For the creation of accurate terrain models with minimal salient points, valid features viewed in several images provide the best map.

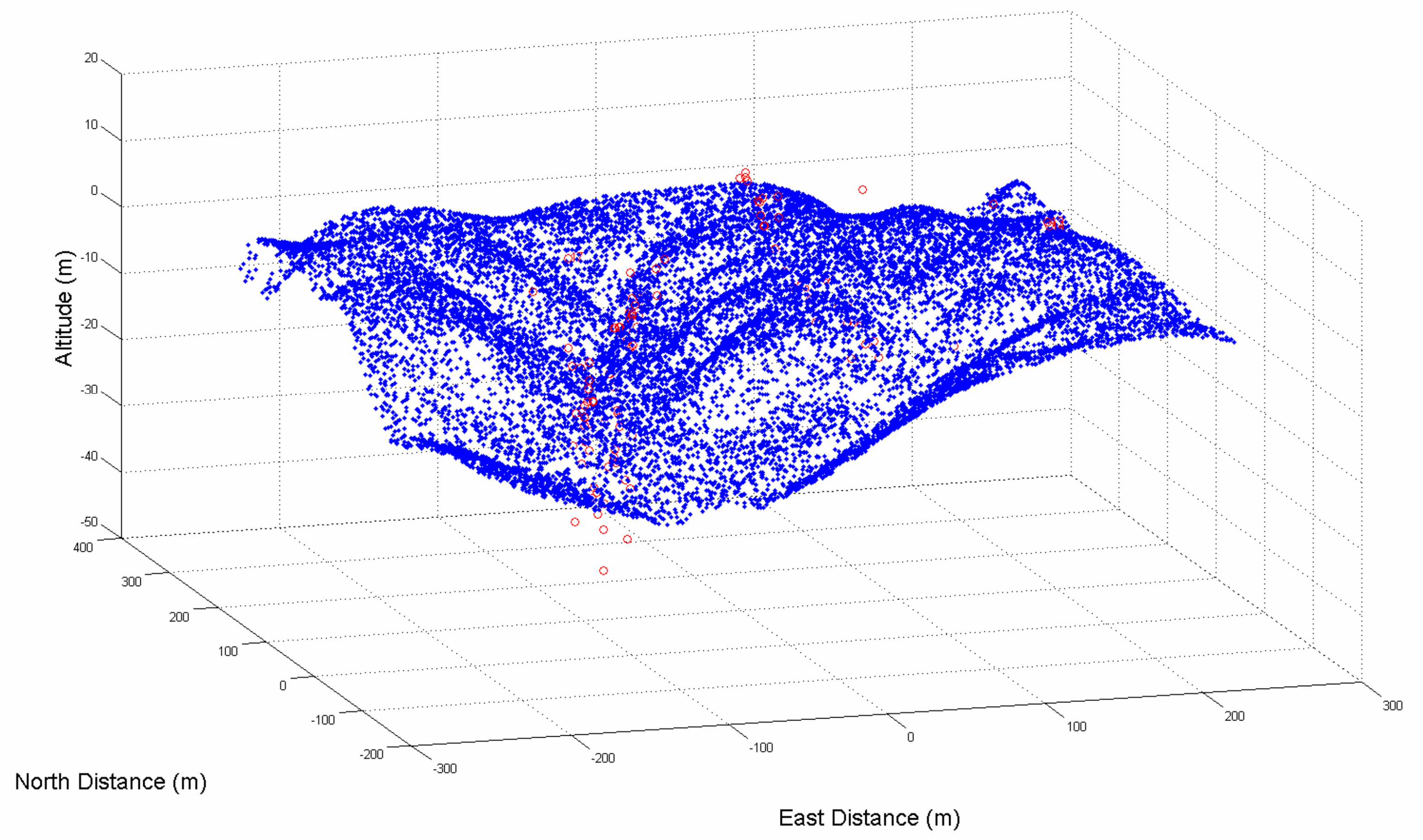


Figure 9-24: 3D LIDAR Data Points and Final Feature Positions

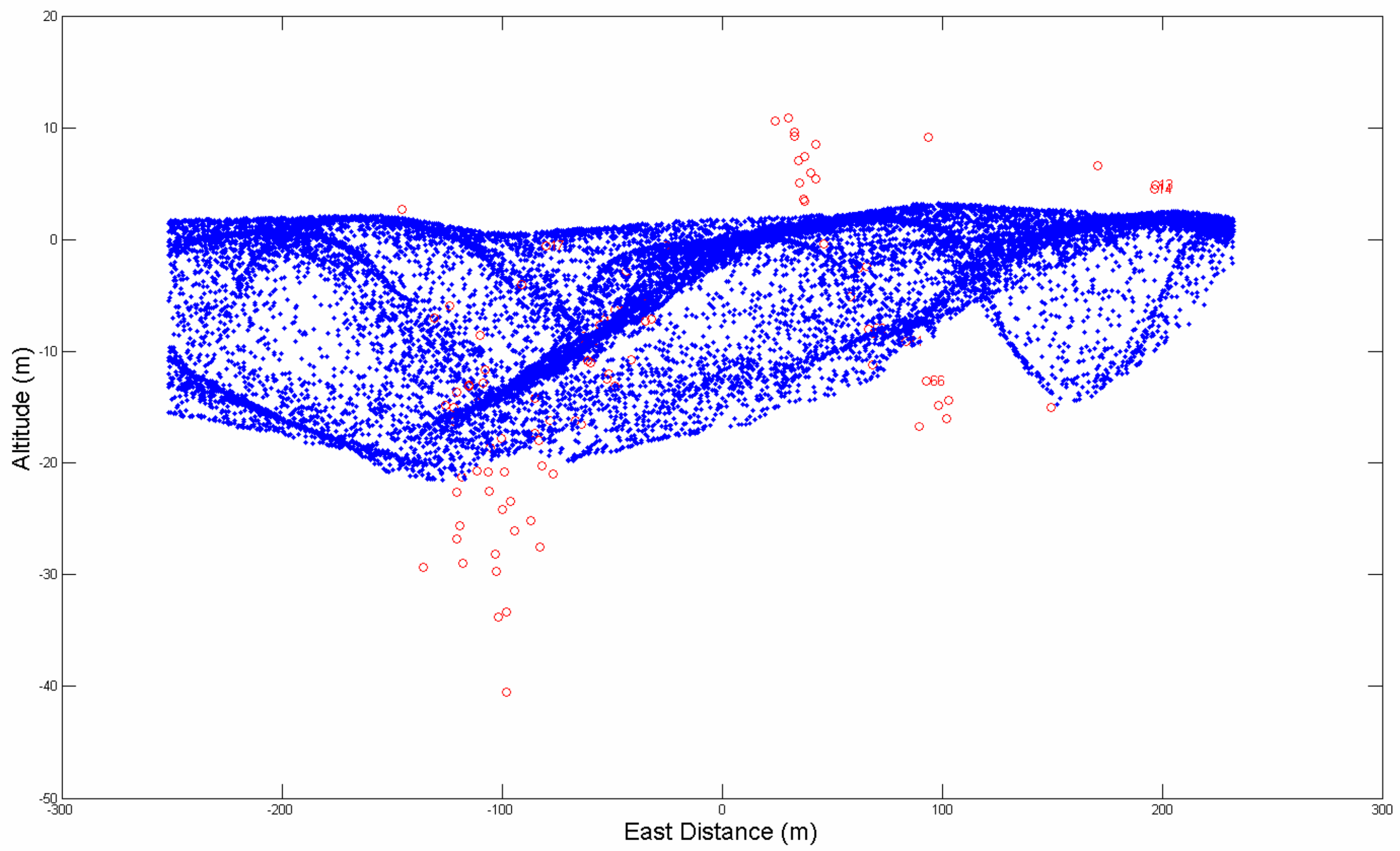


Figure 9-25: LIDAR Data Points and Final East and Altitude Feature Locations

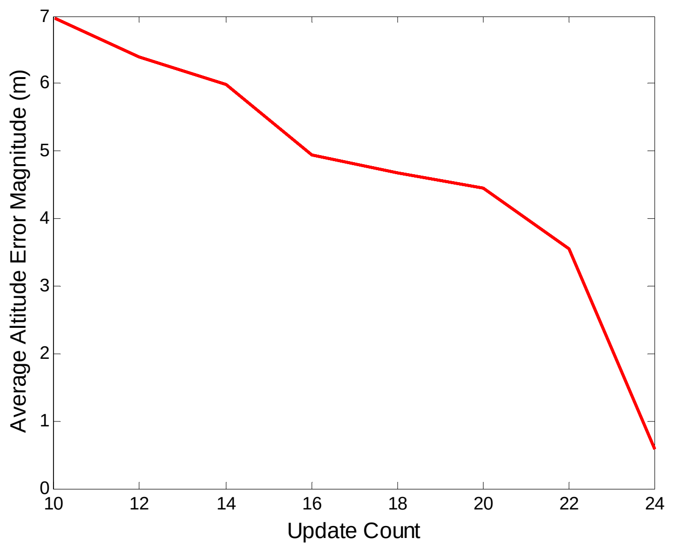


Figure 9-26: Average Altitude Error as a Function of the Number of Correction Updates

Chapter 10: Conclusion and Recommendations

- A novel method has been presented for the tracking of ground targets observed from camera images taken from a moving platform. The observability issues caused by using a monocular camera are addressed by using loosely coupled SLAM and target state estimators and by utilizing the UT to obtain measurement mean and covariance data for the target position as a function of the SLAM states. These issues are also addressed in the same manner by a loosely coupled navigation filter, feature filter, and target filter. Simulation results confirmed that the target states are observable provided the local terrain can be accurately estimated.
- Analysis of experimental data illustrated that loosely coupled aircraft and feature state estimators are capable of accurately determining the latitude and longitude of local terrain features extracted from aerial images. However, the altitude components were more difficult to estimate, emphasizing the need to observe features in several frames with good triangulation characteristics.
- Further research should investigate the use of a gimbaled camera mounted UAV vision system, including possible estimator observability implications. This would allow features to be viewed in several successive frames. Also, practical implementation of the target tracking estimators would require a gimbaled camera due to the generally large discrepancy in UAV and ground target velocities.
- Another method to increase the number of feature observations involves increasing the FOV of the camera lens setup. A larger FOV should provide better triangulations, and hence better altitude estimates.
- A total of three initialization techniques were implemented in simulation. Research confirmed that initialization of feature states from current estimated aircraft states can be detrimental to filter convergence, and thus a loosely coupled filter was used for experimental data analysis. Innovative techniques should be explored for the initialization of features in a manner that is compatible with the SLAM methodology.
- The IMU sensors and GPS module located on the Piccolo II avionics unit are relatively inexpensive and noisy sensors wrought with biases. Including bias states within the

GPS/INS aircraft filter may improve overall filter performance. The GPS estimates may also be improved by utilizing differential corrections from the base station GPS module.

- The Feature Extractor used a three image forward filtering technique to decrease the likelihood of false matches. If false matches become a problem, the filter could readily be modified at the expense of additional computational complexity to require additional matching logic.
- The theoretical target tracking estimator developed and evaluated in simulation should be tested with experimental data, provided the feature altitude estimates can be improved.

References

[1] E. Semerdjiev, L. Mihaylova, and X. Li, "An Adaptive IMM Estimator for Aircraft Tracking," *Proceedings of the International Conference on Information Fusion*, Sunnyvale, CA, July 1999, pp. 770-776.

[2] A. Adrien, D. Filliat, S. Doncieux, and J. Meyer, "2D Simultaneous Localization and Mapping for Micro Air Vehicles," *European Micro Air Vehicle Conference and Flight Competition*, 2006.

[3] J. Kim and S. Sukkarieh, "Airborne Simultaneous Localisation and Map Building," in *Proceedings of the IEEE International Conference on Robotics and Automation,* Taipei, Taiwan, September 2003.

[4] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3. March, 2004.

[5] R. Merwe and E. Wan, "Sigma-Point Kalman Filters for Integrated Navigation," in *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION),* Dayton, OH, June 2004.

[6] B. Stevens and F. Lewis. *Aircraft Control and Simulation*. New Jersey: John Wiley and Sons, Inc., 2003.

[7] M. Yeddanapudi, Y. Bar-Shalom, and K. Pattipati, "IMM Estimation for Multitarget-Multisensor Air Traffic Surveillance," *Proceedings of the IEEE*, vol. 85, no. 1, January 1997, pp. 80-94.

[8] M. Farmer, R. Hsu, and A Jain, "Interacting Multiple Model (IMM) Kalman Filters for Robust High Speed Human Motion Tracking," *Proceedings of the International Conference on Pattern Recognition*, vol. 2, 2002, pp. 20-23.

[9] C. Chen. *Linear System Theory and Design*. 3rd Edition. New York: Oxford University Press, 1999.

[10] F. Lewis and V. Syrmos. *Optimal Control*. 2nd Edition. New York: John Wiley & Sons, Inc., 1995.

[11] R. Nelson. *Flight Stability and Automatic Control*. New York: McGraw-Hill, Inc., 1989.

[12] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, 2007, pp. 1052-1067.

[13] R. Wise, "UAV Control and Guidance for Autonomous Cooperative Tracking of a Moving Target," a dissertation, University of Washington, 2006.

[14] T. Kirubarajan, Y. Bar-Shalom, K. Pattipati, and I. Kadar, "Ground Target Tracking with Variable Structure IMM Estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, January 2000, pp. 26-46.

[15] D. Simon. *Optimal State Estimation*. New Jersey: John Wiley & Sons, Inc., 2006.

[16] J. Guivant and E. Nebot, "Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, June 2001, pp. 242-257.

[17] J. Leonard and P. Newman, "Consistent, Convergent, and Constant-Time SLAM," *Proceedings of the International Joint Conference on Artificial Intelligence*, August 2003.

[18] G. Dissanayake, S. Williams, H. Durrant-Whyte, and T. Bailey, "Map Management for Efficient Simultaneous Localization and Mapping (SLAM)," *Autonomous Robotics*, vol. 12, no. 3, May 2002, pp. 267-286.

[19] Q. Yu and G. Medioni, "Map-Enhanced Detection and Tracking from a Moving Platform with Local and Global Data Association," *IEEE Workshop on Motion and Video Computing*, 2007.

[20] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, January 2004, pp. 91-110.

Appendix A – Quaternion Background Information

This appendix provides quaternion background information. The material presented is obtained from [6]. The reference defines a basic unit quaternion vector as

$$\vec{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$$

where

$$q_0 = \cos(\theta / 2)$$

and

$$\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T = \sin(\theta / 2)\vec{u}\ .$$

$\vec{u}$ describes a unit vector in space which is rotated about by a magnitude Θ to give the new reference frame orientation.

Rotation Matrix from Inertial Frame to Body Frame

$$\mathbf{R}_{\{bi\}} = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix}$$

where

$$\mathbf{R}_{\{ib\}} = \mathbf{R}_{\{bi\}}^T$$

Quaternion to Euler Angle Conversion

$\Phi : \textit{Roll Angle}$

$\Theta : \textit{Pitch Angle}$

$\Psi : \textit{Yaw Angle}$

$$\begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix} = \begin{bmatrix} a\tan\left(\dfrac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} \right) \\ a\sin\left(2(q_0q_2 - q_3q_1) \right) \\ a\tan\left(\dfrac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)} \right) \end{bmatrix}$$

Euler Angle to Quaternion Conversion

$$\vec{q} = \begin{bmatrix} \cos\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right)+\sin\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \sin\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right)-\cos\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right)+\sin\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)\cos\left(\frac{\Theta}{2}\right)\sin\left(\frac{\Psi}{2}\right)-\sin\left(\frac{\Phi}{2}\right)\sin\left(\frac{\Theta}{2}\right)\cos\left(\frac{\Psi}{2}\right) \end{bmatrix}$$

Appendix B – Kalman Filter Algorithm

The following pseudo code is a compact version of the KF algorithm for a linear time invariant system [15].

```
GIVEN
$\hat{\vec{x}}_{0|0}$
$\mathbf{P}_{0|0} = E\left\{\left(\hat{\vec{x}}_{0|0} - \vec{x}_0\right)\left(\hat{\vec{x}}_{0|0} - \vec{x}_0\right)^T\right\}$
$\mathbf{\Phi} = \mathbf{I} + \mathbf{F}T_{\text{int}} + \frac{\mathbf{F}^2 T_{\text{int}}^2}{2}$
$\mathbf{\Gamma} = (\mathbf{I}T_{\text{int}} + \frac{\mathbf{F}T_{\text{int}}^2}{2})\mathbf{G}$
FOR $k = 1, \ldots, \infty$
    $\hat{\vec{x}}_{k|k-1} = \mathbf{\Phi}\hat{\vec{x}}_{k-1|k-1} + \mathbf{\Gamma}\tilde{\vec{u}}_{k-1}$
    $\mathbf{P}^{xx}_{k|k-1} = \mathbf{\Phi}\mathbf{P}^{xx}_{k-1|k-1}\mathbf{\Phi}^T + \mathbf{\Gamma}\mathbf{Q}_{k-1}\mathbf{\Gamma}^T$
    IF (Output Measurement Update)
        $\mathbf{K}_k = \mathbf{P}^{xx}_{k|k-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}^{xx}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k)^{-1}$
        $\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1} + \mathbf{K}_k(\tilde{\vec{z}}_k - \mathbf{H}\hat{\vec{x}}_{k|k-1})$
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1} - \mathbf{K}_k\mathbf{H}\mathbf{P}^{xx}_{k|k-1}$
    ELSE
        $\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1}$
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1}$
    END IF
END FOR
```

Appendix C – Extended Kalman Filter Algorithm

The following pseudo code is a compact version of the EKF algorithm for a nonlinear system [15].

```
**GIVEN**
$\hat{\vec{x}}_{0|0}$
$\mathbf{P}^{xx}_{0|0} = E\left\{\left(\hat{\vec{x}}_{0|0} - \vec{x}_0\right)\left(\hat{\vec{x}}_{0|0} - \vec{x}_0\right)^T\right\}$
**FOR** $k = 1,\ldots,\infty$
    $\hat{\vec{x}}_{k|k-1} = \int_{k-1}^{k} \vec{f}\left(\vec{x} = \hat{\vec{x}}_{k-1|k-1}, \vec{u} = \tilde{\vec{u}}_{k-1}, \vec{w} = \vec{0}\right) dt$
    $\mathbf{F}_k = \left.\frac{\partial \vec{f}}{\partial \vec{x}}\right|_{(\vec{x}=\hat{\vec{x}}_{k-1|k-1,}, \vec{u}=\vec{u}_{k-1})}$
    $\mathbf{G}_k = \left.\frac{\partial \vec{f}}{\partial \vec{w}}\right|_{(\vec{x}=\hat{\vec{x}}_{k-1|k-1,}, \vec{u}=\vec{u}_{k-1})}$
    $\mathbf{\Phi}_k = \mathbf{I} + \mathbf{F}_k T_{\text{int}} + \frac{\mathbf{F}_k^2 T_{\text{int}}^2}{2}$
    $\mathbf{\Gamma}_k = (\mathbf{I} T_{\text{int}} + \frac{\mathbf{F}_k T_{\text{int}}^2}{2})\mathbf{G}_k$
    $\mathbf{P}^{xx}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}^{xx}_{k-1|k-1} \mathbf{\Phi}_k^T + \mathbf{\Gamma}_k \mathbf{Q}_{k-1} \mathbf{\Gamma}_k^T$
    **IF (**Output Measurement Update**)**
        $\mathbf{H}_k = \left.\frac{\partial \vec{h}}{\partial \vec{x}}\right|_{(\vec{x}=\hat{\vec{x}}_{k|k-1})}$
        $\mathbf{K}_k = \mathbf{P}^{xx}_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}^{xx}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
        $\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1} + \mathbf{K}_k\left(\tilde{\vec{z}}_k - \vec{h}\left(\vec{x} = \hat{\vec{x}}_{k|k-1}, \vec{v} = \vec{0}\right)\right)$
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}^{xx}_{k|k-1}$
    **ELSE**
        $\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1}$
        $\mathbf{P}^{xx}_{k|k} = \mathbf{P}^{xx}_{k|k-1}$
    **END IF**
**END FOR**
```