LOW COST/HIGH PRECISION FLIGHT DYNAMICS ESTIMATION USING THE SQUARE-ROOT UNSCENTED KALMAN FILTER

by

Trevor H. Paulsen

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

April 2010

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Trevor H. Paulsen

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	David G. Long, Chair
Date	Richard W. Christiansen
Date	Doran K. Wilde

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Trevor H. Paulsen in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

David G. Long
Chair, Graduate Committee

Accepted for the Department

Michael J. Wirthlin
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of
Engineering and Technology

ABSTRACT

LOW COST/HIGH PRECISION FLIGHT DYNAMICS ESTIMATION USING THE SQUARE-ROOT UNSCENTED KALMAN FILTER

Trevor H. Paulsen

Department of Electrical and Computer Engineering

Master of Science

For over a decade, Brigham Young University's Microwave Earth Remote Sensing (MERS) team has been developing SAR systems and SAR processing algorithms. In order to create the most accurate image reconstruction algorithms, detailed aircraft motion data is essential. In 2008, the MERS team purchased a costly inertial measurement unit (IMU) coupled with high precision global positioning system (GPS) from NovAtel, Inc. In order to lower the cost of obtaining detailed motion measurements, the MERS group decided to build a system that mimics the capability the NovAtel system as closely as possible for a much lower cost. As a first step, the same sensors and a simplified set of flight dynamics are used.

This thesis presents a standalone motion sensor recording system (MOTRON), and outlines a method of utilizing the square-root Unscented Kalman filter (SR-UKF) [1] to estimate aircraft flight dynamics, based on recorded flight data, as an alternative to the extended Kalman filter. While the results of the SR-UKF are not as precise as the NovAtel results, they approach the accuracy of the NovAtel system despite the simplified dynamics model.

ACKNOWLEDGMENTS

I would like to express my gratitude and appreciation to all those who have guided my education.

I would especially like to thank my advisor, friend, and mentor Dr. David Long. He has helped me not only with my research, but with many important decisions I have faced "aLong" the way.

Special thanks go to Dr. Randal Beard for taking time out of his busy schedule to help teach me Kalman filtering basics.

I would also like to express my appreciation towards my colleagues Bryce Ready and Matthew Tolman who spent more time than they probably could afford working with me, and all of my friends in the MERS lab.

Finally, I would like to thank my family for always being encouraging and supportive. Particularly, I thank my wonderful wife, Caryn, who has been so patient with me through this whole process.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Purpose

Synthetic aperture radar (SAR) is a valuable tool with applications ranging from military to environmental. There are many types of SAR, but each relies on special processing techniques to form useful imagery. These processing techniques require accurate measurements of aircraft speed and attitude to correct aberrations within SAR images. For over a decade, the Microwave Earth Remote Sensing (MERS) team at Brigham Young University (BYU) has been developing SAR instruments and processing algorithms that incorporate these motion measurements. With the addition of BYU's nu-SAR system in 2008, an accurate standalone motion measurement system was needed.

In 2008, BYU purchased a costly motion dynamics estimation system from NovAtel, Inc. The system utilizes three orthogonal accelerometers and laser gyroscopes coupled with a high precision GPS system. It also includes a CPU that processes the sensor and GPS data into a high precision navigation solution consisting of attitude and position information. In order to decrease the dependence on expensive motion measurement systems, BYU's MERS team determined a standalone recording/processing system was necessary and also wanted to develop processing algorithms that mimic the capability of a high-end navigation system.

The purpose of this thesis is to present a system to couple high precision accelerometers and laser gyroscopes with a high precision GPS system into an accurate, onboard motion measurement system using the square-root Unscented Kalman filter (SR-UKF) presented by Merwe [1]. Specifically, this thesis addresses a motion recording onboard device (MOTRON) used to record and time sync GPS, accelerometer,

and gyroscope measurements, as well as SR-UKF techniques to optimally estimate aircraft attitude, position, and velocity from these measurements.

This thesis demonstrates that the SR-UKF can be a very effective tool for aircraft dynamics estimation. Although the NovAtel processing system is much more sophisticated, the SR-UKF presented in this thesis approaches the effectiveness of the NovAtel system. While it is expected that the techniques described by this thesis will not outstrip the performance of a $40,000 motion measurement unit, it is shown that with creative methodology, a much less complicated system implementing an effective type of Kalman filter can perform quite well at a lower cost.

1.2 Contributions

This thesis contributes to the base of estimation theory by presenting a standalone sensor recording scheme that can be used in any fixed-wing aircraft platform. This thesis further contributes by applying the SR-UKF to flight dynamics and motion estimation. Previously, the extended Kalman filter (EKF) has been used with these sensors by NovAtel [2] to estimate motion measurements. Comparison of the NovaTel filter and the SR-UKF is also presented. This analysis gives valuable insight into the functionality of the SR-UKF for flight dynamics estimation.

1.3 Outline

The remainder of this thesis is organized into the following chapters:

- Chapter 2 gives a background on the need for motion measurement systems within SAR processing algorithms, a brief survey of aircraft flight dynamics, and a presentation of the SR-UKF.

- Chapter 3 presents a motion sensor recording scheme in order to collect data from laser gyroscopes, accelerometers, and high precision GPS. It introduces the requirements the MOTRON must fulfill, and how the MOTRON meets those requirements.

- Chapter 4 outlines the specific implementation of the SR-UKF to the MERS aircraft flight dynamics estimation problem. This chapter also gives specific approximations that are used by the SR-UKF to simplify calculations.

- Chapter 5 presents the results of the SR-UKF on simulated data and on actual recorded data. In the simulated case, the output of the SR-UKF is compared to the truth data provided by the simulator, and in the actual flight, the SR-UKF solution is compared to the NovAtel solution.

- Chapter 6 concludes the thesis and suggests areas in which further work can be done to refine the SR-UKF implementation.

Chapter 2

Background

2.1 Introduction

In order to create a motion tracking system, it is important to look at why such a system is needed. This chapter discusses the need for a motion tracking system, reviews basic aircraft dynamics, and provides a formulation of a nonlinear square-root "unscented" Kalman filter. This chapter also outlines assumptions made in flight dynamics in order to simplify state space calculations.

2.2 SAR Image Processing

Microwave remote sensing, particularly Synthetic Aperture Radar (SAR), has been in use since the 1960s. Some of the significant advantages of SAR over conventional aerial photography are its ability to penetrate clouds, independence of the sun as a source of illumination, and the ease at which man-made objects can be recognized. Often, the combined use of SAR and photography allow a greater study of geometric and molecular properties of a surface. SAR systems are very complex devices, and as such, only a brief overview of SAR systems is given here.

2.2.1 SAR System Overview

In a typical SAR application, the SAR antenna is attached to the side of an aircraft. Once the aircraft has achieved level flight, a pulse from the antenna is transmitted to the ground below. Because diffraction requires a large antenna to produce a narrow beam, a single pulse from the radar is rather broad; often illuminating the terrain from directly beneath the aircraft to the horizon. If the topography is approximately flat, the times at which that echo returns allow objects at different distances

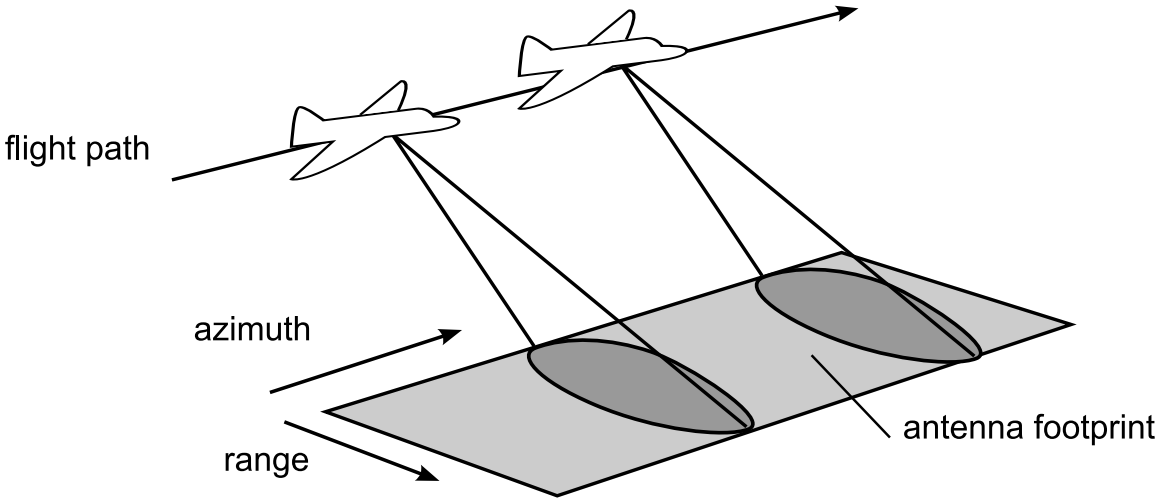

Figure 2.1: SAR uses multiple antenna pulses to create a synthetic antenna. By combining each pulse return from an area, high resolution images can be created.

to be distinguished. If the amplitude and phase of the return signal from a piece of ground are recorded, and the aircraft emits a series of these pulses as it travels, the results of these pulses can be combined to make a much higher resolution image. In effect, the series of observations can be joined together just as if they had all been made concurrently from a very large antenna. This process creates a synthetic aperture much larger than the length of the antenna or aircraft. Image resolution of SAR is proportional to the radio signal bandwidth, and depends upon the post-processing techniques used. Constructing a SAR image usually requires significant computational resources. Often, image construction is done on the ground well after the flight has been made.

2.2.2 The Need for Motion Compensation

Because synthetic aperture processing usually depends upon the assumption that the radar is traveling horizontally in a straight line at constant speed, any movement outside of that line is not ideal. Errors in acceleration or attitude in an aircraft's flight can cause serious problems for the SAR. Consider the example illustrated by Fig. 2.2. Because the acceleration and attitude are non-ideal, the radar return is not what it is desired to be. As a result, the image is defocused and displaced. According

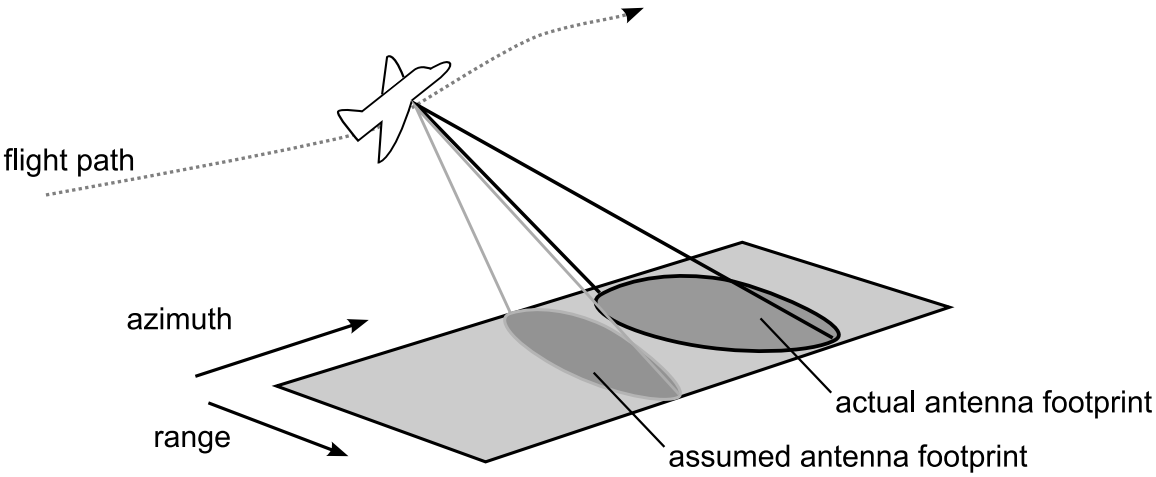

Figure 2.2: Any non-ideal motion in an aircraft can cause serious problems for synthetic aperture processing.

to Ulaby [3], the bounds on the size of the error velocities u_y (height) and u_z (side to side) are given by

$$u_y \sin\theta - u_z \cos\theta \leq \lambda f_{D0} \epsilon_r, \tag{2.1}$$

where θ is the angle between the vertical and radar beam such that $\cos\theta = \frac{\text{height}}{\text{range}}$, λ is the wavelength of the beam, f_{D0} is the desired Doppler frequency and

$$\frac{f_{D\epsilon}}{f_{D0}} \leq 2\epsilon_r, \tag{2.2}$$

where $f_{D\epsilon}$ is the Doppler error component and ϵ_r is a limit bound allowed in the Doppler frequency. Considering an example aircraft with $\epsilon_r = 0.1$, $f_{D0} = 1$ Hz, and $\lambda = 3$ cm, the limit is found to be

$$u_y \sin\theta - u_z \cos\theta \leq 3\text{mms}^{-1}. \tag{2.3}$$

In addition, the errors in acceleration must also be compensated. This essentially means compensating for changes in velocity. While fixed-velocity errors affect the presumed direction of the radar beam, acceleration errors affect the focus of the beam because varying Doppler shift causes the desired signal to drift out of the Doppler

filter band. Using the same example aircraft it can be shown that

$$a_y \sin\theta - a_z \cos\theta \leq 0.006\text{m/s}^2, \tag{2.4}$$

meaning that any acceleration error must be within 6.1×10^{-4}g, which is extremely sensitive.

Furthermore, an aircraft may roll, pitch, and yaw. Each of these movements cause variation in the SAR return which can lead to errors in image reconstruction. Rolling alters the gain for a particular point on the ground. If the antenna rolls too much, this can be a serious problem, but it is not as severe as any yawing or pitching. If an aircraft yaws or pitches while recording SAR data, it moves the illuminated area away from a side-looking direction to a direction that is slightly ahead of or behind the plane. This causes distortions in the image, and shifts the Doppler frequencies away from the broadside of the plane.

With tiny attitude and velocity errors causing serious problems for the SAR, it becomes absolutely necessary to compensate for these errors. It is unrealistic to think that an aircraft can fly such a straight course. This is why great care is taken to measure non-ideal aircraft movement so that the unexpected motion can be compensated for in processing algorithms.

2.3 Aircraft Flight Dynamics

In order to most effectively find inconsistencies in an aircraft's flight path, it is important to understand the ways in which an aircraft's position can be represented. This section briefly presents some basics behind coordinate frames, kinematics, dynamics, and nonlinear equations of motion. The notation used here is that used by Beard in [4]; however, this notation is common to most aeronautics literature.

2.3.1 Coordinate Frames

There are various coordinate systems that can be used to describe the position, velocity, and orientation of aircraft. Each of these coordinate systems can be

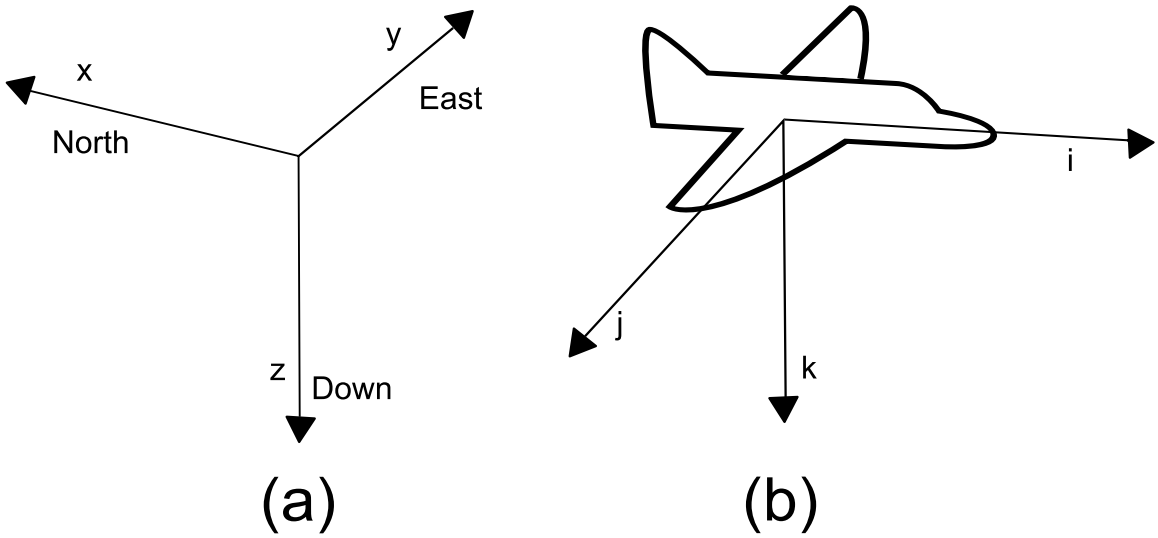

Figure 2.3: Two of the most important frames of orientation. (a)The world frame. (b)The body frame.

transformed into the frame of another. There are several reasons that multiple coordinate frames need to be used. First, Newton's equations of motion are given in the coordinate frame attached to the aircraft. Second, aerodynamic forces are applied in the body frame of the aircraft. Third, onboard sensors measure information with respect to the body frame, while outside sensors (such as GPS or RADAR) measure information in the world frame. Because all of this information needs to be incorporated, it is necessary to be able to express any measurement in terms of any other coordinate system.

One coordinate frame is transformed into another through two basic operations: rotation and translation. Rotations can be performed using a rotation matrix. A rotation matrix is an n-dimensional square matrix that acts as a rotation of Euclidean space. Two aircraft coordinate frames, the world frame and the body frame, are related to each other through a rotation.

When rotated through three Euler angles, roll, pitch, and yaw, it is possible to rotate from the body frame to the word frame and vice versa. However, a single transformation through these three angles is not always unique. In order to create a unique transformation, it is necessary to restrict the order of these Euler rotations.

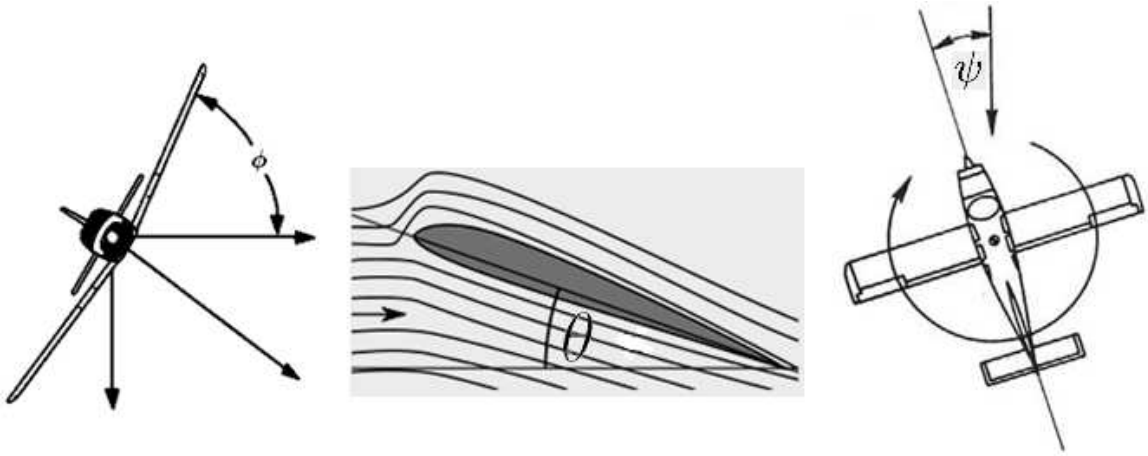

Figure 2.4: The three Euler angles. ϕ is roll, θ is pitch, and ψ is yaw. Each follow the right hand rule about the body frame coordinate axis.

Therefore, to reach the world frame from the body frame, we construct a rotation matrix by first rolling, then pitching, then yawing. This is equivalent to rotating three times, or multiplying three rotation matrices.

$$\begin{aligned}\boldsymbol{R}_{BtoW} &= \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \\ &= \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \end{aligned} \tag{2.5}$$

where ϕ, θ, and ψ represent roll, pitch, and yaw respectively, and $c\phi = \cos\phi$, and $s\theta = \sin\theta$. Similarly, transforming from the world to body frame requires the same Euler angles, but they must be applied in reverse order. This is equivalent to multiplying by the transpose of $\boldsymbol{R}_{BtoW}$.

$$\boldsymbol{R}_{WtoB} = \boldsymbol{R}_{BtoW}^{T} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta c\psi & c\phi c\theta \end{bmatrix}. \tag{2.6}$$

Converting angular rates (p, q, and r) into the world frame ($\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$) is also important. By recognizing that

$$\begin{aligned}\begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \end{aligned} \tag{2.7}$$

as shown by Beard in [4] and inverting,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{2.8}$$

we can express changes in angular rates within the body frame as changes in the Euler angles. With these rotation matrices we can represent all of the sensor outputs we receive in terms of one another. This is crucial for extracting as much information as possible from the sensors.

It is very important to understand what coordinate frame the measurements are taken in. Position measurements are almost always taken from the world frame via GPS. Velocity measurements can be expressed in the world frame (north, east, down velocity) or in the body frame (velocity in the i, j, k directions). Acceleration can be represented in further complicated frames. This is due to the fact that every time we differentiate a measurement, it must be differentiated with respect to a coordinate frame. For example, differentiating body frame velocity with respect to the world frame gives non-centripetal acceleration. Differentiating the body frame velocity with respect to the body frame gives an acceleration measurement that includes centripetal acceleration. In addition, angular rates measured by gyroscopes are measured in the body frame, while Euler angles are measured in the world frame.

2.3.2 Characteristics of Aircraft Motion

With an understanding of coordinate frames, a brief discussion of aircraft flight characteristics is warranted. First, one of the most important characteristics of an airplane is understanding how it turns. When an aircraft turns, it rolls to accomplish a change in heading. This is known as a coordinated turn. Note that if the roll angle is zero, the change in world frame heading is related only to the wind and sideslip angle. If sideslip and wind are neglected, then the roll angle is zero if there is no change in heading. Under these conditions, a coordinated turn can be modeled by

$$\dot{\psi} = \frac{g}{v} \tan \phi, \tag{2.9}$$

where $\dot{\psi}$ is the change in heading, g is the acceleration due to gravity, ϕ is the roll angle, and v is the total velocity of the aircraft. Furthermore, we can model the body frame velocity according to

$$\begin{aligned} u &= v \cos \alpha \cos \beta, \\ v &= v \sin \beta, \\ w &= v \sin \alpha \cos \beta, \end{aligned} \tag{2.10}$$

where u, v, and w are the body frame velocity in the i, j, and k directions respectively. v is the total velocity, α represents the angle of attack, and β represents the sideslip angle. Neglecting sideslip also allows us to assume that all of the velocity present in the body frame of the aircraft is present in the i and k directions. This means that

$$\begin{aligned} u &= v \cos \alpha, \\ v &= 0, \\ w &= v \sin \alpha. \end{aligned} \tag{2.11}$$

Assuming sideslip is zero is an approximation, but for larger aircraft with a large rudder, it can be a good approximation. Assuming no wind is a worse approxi-

mation, but if we assume that SAR readings are made on sufficiently calm days, then it can be acceptable. These assumptions greatly simplify the dynamics and help us to formulate a simpler state space model for the Kalman filter.

2.4 The Kalman Filter

The Kalman filter is a recursive estimator used to estimate the state of a linear time-varying state equation, in which the states are driven by noise and observations are made in the presence of noise. There are many variations of the Kalman filter, particularly when extending this idea to non-linear models. This section addresses the basics of Kalman filtering and useful extensions of the idea using the Unscented Kalman Filter (UKF) and it's square-root variation for numerical stability. The notation used in this section for the Kalman filter is that of Moon [5].

2.4.1 The State-Space Model

The Kalman filter is a method of recursive and sequential estimation based on a state space model,

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{F}_t \boldsymbol{x}_t + \boldsymbol{\xi}_t, \\ \boldsymbol{y}_t &= \boldsymbol{H}_t \boldsymbol{x}_t + \boldsymbol{\eta}_t, \end{aligned} \tag{2.12}$$

where $\boldsymbol{F}_t$ represents the state transition model, $\boldsymbol{H}_t$ is the observation model, $\boldsymbol{\xi}_t$ is the state process noise, and $\boldsymbol{\eta}_t$ is the observation noise in vector form. For simplicity, all processes are assumed to be real. The process noise is assumed to be zero mean, with covariance

$$E\boldsymbol{\xi}_t \boldsymbol{\xi}_\tau^T = \boldsymbol{Q}_t \delta_{t,\tau}, \tag{2.13}$$

and is assumed to be uncorrelated among samples. The observation noise $\boldsymbol{\eta}_t$ is also assumed to be zero mean, uncorrelated, with covariance

$$E\boldsymbol{\eta}_t \boldsymbol{\eta}_\tau^T = \boldsymbol{R}_t \delta_{t,\tau}. \tag{2.14}$$

We assume that the process and obervation noise are uncorrelated as well. Finally, there is some initial condition (or priori density) on the random variable denoted $\boldsymbol{x}_0$.

2.4.2 Formulation of the Kalman Filter

There are several ways to approach the Kalman filter. The method described here is often referred to as the "innovations approach". The Kalman filter has two important steps, the time-update and the measurement-update. The time-update is performed using

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1|t} &= \boldsymbol{F}_t \hat{\boldsymbol{x}}_{t|t}, \\ \boldsymbol{P}_{t+1|t} &= \boldsymbol{F}_t \boldsymbol{P}_{t|t} \boldsymbol{F}_t^T + \boldsymbol{Q}, \end{aligned} \tag{2.15}$$

and the measurement update is performed using

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1|t+1} &= \hat{\boldsymbol{x}}_{t+1|t} + \boldsymbol{K}_{t+1}[\boldsymbol{y}_{\text{measured}} - \boldsymbol{H}_{t+1} \hat{\boldsymbol{x}}_{t+1|t}], \\ \boldsymbol{P}_{t+1|t+1} &= (\boldsymbol{I} - \boldsymbol{K}_{t+1} \boldsymbol{H}_{t+1}) \boldsymbol{P}_{t+1|t}, \end{aligned} \tag{2.16}$$

where

$$\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1|t} \boldsymbol{H}_{t+1}^T [\boldsymbol{H}_{t+1} \boldsymbol{P}_{t+1|t} \boldsymbol{H}_{t+1}^T + \boldsymbol{R}]^{-1}, \tag{2.17}$$

and $\boldsymbol{P}$ is the state covariance matrix and $\boldsymbol{Q}$ and $\boldsymbol{R}$ have been defined previously.

The Kalman filter is a solution to the general problem of state estimation on a linear system. When using a Kalman filter on linear Gaussian systems, it is an optimal estimator. This is due to the fact that Gaussian distributions are completely described by a mean and covariance. Unfortunately, this derivation of the Kalman filter is only useful for linear systems. To be used with aircraft dynamics, a different approach to the Kalman filter must be taken.

2.4.3 Non-Linear Extensions of the Kalman Filter

Consider a general nonlinear system of the form

$$\begin{aligned} \boldsymbol{x}_{t+1} &= f(\boldsymbol{x}_t, t) + \boldsymbol{\xi}_t, \\ \boldsymbol{y}_t &= h(\boldsymbol{x}_t, t) + \boldsymbol{\eta}_t, \end{aligned} \tag{2.18}$$

where f and h are nonlinear functions of $\boldsymbol{x}_t$ and t.

The general nonlinear estimation problem is extremely difficult, and no general solution to the general nonlinear filter problem is available. One reason the linear problem is more convenient to solve is that when the noise and initial conditions are Gaussian, the state, $\hat{\boldsymbol{x}}_t$, is guaranteed to be Gaussian as well. However, if f or h is nonlinear, then $\hat{\boldsymbol{x}}_{t|\tau}$ is not guaranteed to be Guassian, so either a linearization is required, or there must be a different approach to estimating the state error covariance. A typical approach to solving this problem is to linearize the nonlinear function f using the Jacobian,

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_L} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_L}{\partial x_1} & \frac{\partial f_L}{\partial x_2} & \cdots & \frac{\partial f_L}{\partial x_L} \end{bmatrix}, \tag{2.19}$$

where L is the dimension of the state. With this approach, we can then use the Jacobian of f in place of the matrix F of the linear Kalman filter, but updated at every sample. Similarly, a linearization of h is needed, so by also computing its Jacobian we can use the Jacobian of h in place of matrix H. This is generally referred to as the Extended Kalman Filter (EKF).

Another approach to the nonlinear problem is the Unscented Kalman Filter (UKF) presented by Julier and Uhlmann [6]. The UKF uses a deterministic approach to calculate mean and covariance terms. Basically $2L + 1$ "sigma" points are chosen based on a square-root decomposition of the prior covariance. Each of these sigma points are then passed to the nonlinear function f without approximation. A weighted mean and covariance is then taken from this new set of points. Julier and Uhlmann

[6] have shown that this approach results in approximations accurate to the third order (Taylor series expansion) for Gaussian inputs for all nonlinearities. For non-Gaussian inputs, approximations are accurate to at least the second order. The EKF's linearization approach, on the other hand, results only in first-order accuracy.

The UKF chooses sigma points using the following formulas:

$$\boldsymbol{\chi}_t = \left[\begin{array}{ccc} \hat{\boldsymbol{x}}_t & \hat{\boldsymbol{x}}_t + \eta\sqrt{\boldsymbol{P}_t} & \hat{\boldsymbol{x}}_t - \eta\sqrt{\boldsymbol{P}_t} \end{array} \right], \tag{2.20}$$

and

$$\eta = \sqrt{L\alpha^2}, \tag{2.21}$$

where $\boldsymbol{\chi}_t$ is a set of $2L + 1$ sigma points and α is a constant that determines the spread of the sigma points around $\hat{\boldsymbol{x}}$ and is usually set to $0.0001 \leq \alpha \leq 1$. It also uses the following weights:

$$\begin{aligned} W_0^{(m)} &= \frac{\alpha^2 - 1}{\alpha^2}, \\ W_0^{(c)} &= \frac{\alpha^2 - 1}{\alpha^2} + (1 - \alpha^2 + \beta), \\ W_i^{(m)} = W_i^{(c)} &= \frac{1}{2L\alpha^2}, \end{aligned} \tag{2.22}$$

where $\beta = 2$ if the system is Guassian. Finally, with these weights in place, we can calculate the state mean and covariance time update using a deterministic approach,

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1|t} &= \sum_{i=0}^{2L} W_i^{(m)} \boldsymbol{\chi}_{i,t+1|t}, \\ \boldsymbol{P}_{t+1|t} &= \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\chi}_{i,t+1|t} - \hat{\boldsymbol{x}}_{t+1|t}][\boldsymbol{\chi}_{i,t+1|t} - \hat{\boldsymbol{x}}_{t+1|t}]^T + \boldsymbol{Q}, \end{aligned} \tag{2.23}$$

where $\boldsymbol{\chi}_{i,t+1|t}$ represents the sigma points which have been passed through the nonlinear function f. The measurement update is performed similarly,

$$\begin{aligned}
\boldsymbol{P}_{yy} &= \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\mathcal{Y}}_{i,t+1|t} - \hat{\boldsymbol{y}}_{t+1}][\boldsymbol{\mathcal{Y}}_{i,t+1|t} - \hat{\boldsymbol{y}}_{t+1}]^T + \boldsymbol{R}, \\
\boldsymbol{P}_{xy} &= \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\chi}_{i,t+1|t} - \hat{\boldsymbol{x}}_{t+1|t}][\boldsymbol{\mathcal{Y}}_{i,t+1|t} - \hat{\boldsymbol{y}}_{t+1}]^T, \\
\boldsymbol{\mathcal{K}}_{t+1} &= \boldsymbol{P}_{xy} \boldsymbol{P}_{yy}^{-1},
\end{aligned} \tag{2.24}$$

where $\boldsymbol{\mathcal{K}}$ denotes the Kalman gain for the UKF. $\boldsymbol{\mathcal{Y}}_{i,t+1|t}$ is a new set of sigma points, specifically the points from $\boldsymbol{\chi}_{i,t+1|t}$, which have been passed through the nonlinear function h. This approach is different from the EKF in that it does not use a Jacobian, but instead, uses a deterministic sampling to manually approximate the state covariance. The computation involved is the same order of magnitude according to Merwe [1]. This is due to the fact that the Jacobians are removed, but they are replaced with deterministic outer products. The final step of the measurement update is performed analogously to the linear Kalman filter:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{t+1|t+1} &= \hat{\boldsymbol{x}}_{t+1|t} + \boldsymbol{\mathcal{K}}_{t+1}(\boldsymbol{y}_{\text{measured}} - \hat{\boldsymbol{y}}_{t+1}), \\
\boldsymbol{P}_{t+1|t+1} &= \boldsymbol{P}_{t+1|t} - \boldsymbol{\mathcal{K}}_{t+1} \boldsymbol{P}_{yy} \boldsymbol{\mathcal{K}}_{t+1}^T.
\end{aligned} \tag{2.25}$$

2.4.4 Square-Root Implementation of the UKF

Because Kalman filter equations tend to be somewhat poorly conditioned numerically, many algorithms have been developed to address this problem which have better numerical conditioning. One of these algorithms, known as the square-root Kalman filter, propagates $\boldsymbol{S}_{t+1}$ which is the Cholesky square-root of the matrix $\boldsymbol{P}_{t+1|t}$. Merwe [1] has proposed an efficient method for implementing this modification to the UKF known as the SR-UKF. The SR-UKF makes use of three linear algebra techniques: QR decomposition, Cholesky factor updating, and efficient least squares.

First, sigma points must be calculated in a slightly different manner,

$$\boldsymbol{\chi}_t = \begin{bmatrix} \hat{\boldsymbol{x}}_t & \hat{\boldsymbol{x}}_t + \eta \boldsymbol{S}_t & \hat{\boldsymbol{x}}_t - \eta \boldsymbol{S}_t \end{bmatrix}, \tag{2.26}$$

because we are now utilizing $\boldsymbol{S}_t$ instead of $\boldsymbol{P}_t$. $\boldsymbol{\chi}_t$ is therefore computed the same way as before, and the mean is also calculated in the same way. The covariance time update must now be performed using the following two steps:

$$\begin{aligned} \boldsymbol{S}_{t+1|t} &= \text{qr} \begin{bmatrix} \sqrt{W_i^{(c)}} & (\boldsymbol{\chi}_{1:2L,t+1|t} - \hat{\boldsymbol{x}}_{t+1}) & \sqrt{\boldsymbol{Q}} \end{bmatrix}, \\ \boldsymbol{S}_{t+1|t} &= \text{cholupdate} \begin{bmatrix} \boldsymbol{S}_{t+1}, & \boldsymbol{\chi}_{0,t+1} - \hat{\boldsymbol{x}}_{t+1}, & W_0^{(c)} \end{bmatrix}. \end{aligned} \tag{2.27}$$

In the measurement update, the equation for $\boldsymbol{P}_{yy}$ is replaced by the measurement update equation for $\boldsymbol{S}_{y_{t+1}}$,

$$\begin{aligned} \boldsymbol{S}_{y_{t+1}} &= \text{qr} \begin{bmatrix} \sqrt{W_i^{(c)}} & (\boldsymbol{\mathcal{Y}}_{1:2L,t+1|t} - \hat{\boldsymbol{y}}_{t+1}) & \sqrt{\boldsymbol{R}} \end{bmatrix}, \\ \boldsymbol{S}_{y_{t+1}} &= \text{cholupdate} \begin{bmatrix} \boldsymbol{S}_{y_{t+1}}, & \boldsymbol{\mathcal{Y}}_{0,t+1} - \hat{\boldsymbol{y}}_{t+1}, & W_0^{(c)} \end{bmatrix}, \end{aligned} \tag{2.28}$$

and the equation for $\boldsymbol{\mathcal{K}}_{t+1}$ becomes (using MATLAB's efficient least squares algorithm implemented with the '/' operator),

$$\boldsymbol{\mathcal{K}}_{t+1} = (\boldsymbol{P}_{xy} / \boldsymbol{S}_{y_{t+1}}^T) / \boldsymbol{S}_{y_{t+1}}, \tag{2.29}$$

where $\boldsymbol{S}_{y_{t+1}}^T \boldsymbol{S}_{y_{t+1}}$ takes the place of $\boldsymbol{P}_{yy}$, and $\boldsymbol{S}_{t+1}$ is updated by

$$\begin{aligned} \boldsymbol{U} &= \boldsymbol{\mathcal{K}}_{t+1} \boldsymbol{S}_{y_{t+1}}, \\ \boldsymbol{S}_{t+1|t+1} &= \text{cholupdate} \begin{bmatrix} \boldsymbol{S}_{t+1|t}, & \boldsymbol{U}, & -1 \end{bmatrix}. \end{aligned} \tag{2.30}$$

This square-root implementation has significant advantages over the UKF. According to Merwe [1], the SR-UKF is an $\mathcal{O}(L^2)$ algorithm as opposed to $\mathcal{O}(L^3)$ for the UKF. This is due to the fact that the SR-UKF takes advantage of many optimal matrix algorithms. Merwe states that the SR-UKF is about 20% faster than the UKF

in experimental results. With the SR-UKF in place, we are prepared to apply it to the flight dynamics problem for SAR image processing.

2.5 Conclusion

This chapter has presented the need for motion measurement, a basis for understanding aircraft flight dynamics, and a formulation of the Kalman filter. With this information, a state estimation scheme can be readily initiated.

Chapter 3

Motion Recording Onboard Device (MOTRON)

When the BYU MERS lab purchased a motion sensing system from NovAtel Inc., a compact data collection and storage device for this system became necessary. As a consequence, I created the MOTRON (MOTion Recording ONboard device). The goal of the MOTRON system is to create a small, lightweight, user-friendly, and efficient data collection device. Specifically, the MOTRON collects data from a Novatel Propak V3 system. This system includes an IMU (Inertial Measurement Unit) consisting of three finely calibrated fiber optic gyros, three highly sensitive accelerometers, and a high precision GPS system.

In order to create a motion tracking system, a data collection device is needed to accumulate all of the necessary measurements for that system. This chapter outlines the requirements that the MOTRON fulfills, the specifications of the sensors to be measured, the components and operation of the MOTRON, and finally, examples of data that have been captured using the MOTRON and how to make the data useful for analysis.

3.1 Requirements for the MOTRON

The MOTRON is designed to support Synthetic Aperture Radar (SAR) systems. A typical SAR system amasses enormous amounts of radar data that are later processed to produce images. It is useful to have a separate motion measuring system to remove any unnecessary computation and memory usage that may further burden a SAR system. It is also useful to collect motion data independent of radar. Because there are no engineers onboard the aircraft when it collects radar data, the MOTRON must be very easy to use. The MOTRON must be able to collect data with little

or no human interaction during the flight and have a very user friendly interface for data extraction and set up.

Because the MOTRON is meant to be mounted to an aircraft, it is important that its weight and size be minimized. This also makes it a viable data capturing option for future, smaller aircraft. Since the sensors are independent of the MOTRON, the MOTRON must effectively link to those sensors without losing any data. This means that the data transfer rate must be sufficient to handle all of the incoming data. Finally, the MOTRON must have an efficient way to store all of the sensor data. It is important that this data is easy to extract once recorded and readily available.

3.2 Motion Sensors

Given the requirements of the MOTRON, this section considers the way that the MOTRON connects to each of the outside sensors, specifically, the NovAtel ProPak-V3 in conjunction with an Inertial Measurement Unit (IMU), GPS antenna, the SAR system, and any additional sensors. A schematic is shown in Fig. 3.1. The next subsections include information about the various sensors used with the MOTRON.

3.2.1 NovAtel ProPak V3 with IMAR IMU

NovAtel's ProPak-V3 is a high-performance GPS receiver with 72 available channels, L1 and L2 GLONASS, USB communication and SPAN support. Figure 3.2 illustrates the different connectors to the ProPak. The ProPak connects to the IMU, MOTRON, SAR System (for data syncing), and to the GPS Antenna. This system outputs GPS and inertial measurements through the USB-serial port to the MOTRON. It also outputs a time stamp to the SAR system in order to time sync the data in post processing.

The ProPak connects to the antenna shown in Fig. 3.3. The GPS-701-GG antenna is part of NovAtel's GPS-700 antenna series and offers access to GPS L1 and L2 frequencies, as well as GLONASS. For added accuracy, the antenna phase center remains constant as the azimuth and elevation angle of GPS satellites change. Since

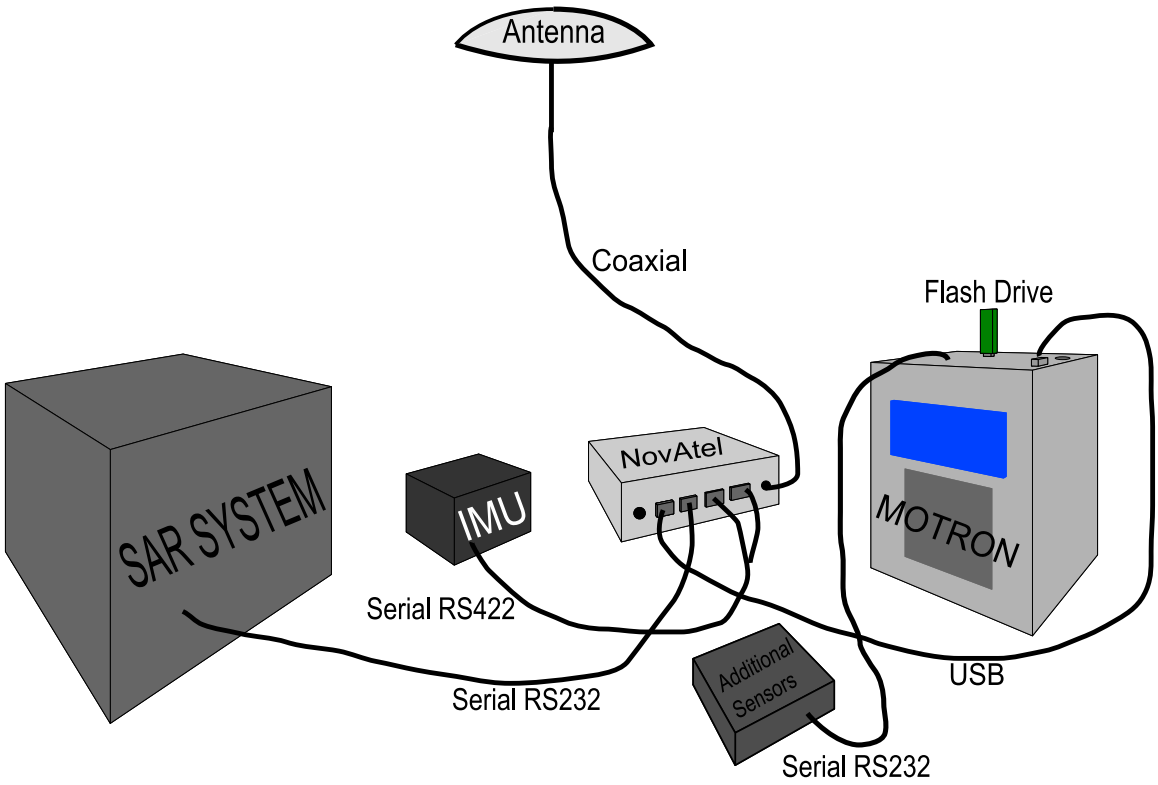

Figure 3.1: Schematic showing how the MOTRON, SAR system, NovAtel, IMU, Antenna, Flash Drive, and any additional sensors are connected to each other.

signal reception is unaffected by the rotation of the antenna or GPS satellite elevation, placement and installation of the antenna is easy. It connects to the ProPak using a standard coaxial cable shown in Fig. 3.2 and is permanently attached to the aircraft.

The third component of the NovAtel system is the Inertial Measurement Unit (IMU). The particular IMU that came with the system is an IMAR iIMU-FSAS tactical grade IMU. The iIMU-FSAS is a relatively small size IMU consisting of three fiber optic rate gyros of class 0.75 deg/hr and three servo-accelerometers of class 1 mg. This unit is pictured in Fig. 3.4. It is designed for ruggedized applications and is internally equipped with shock absorbers. It communicates with ProPak via RS422 on an HDLC protocol.

After each component has been connected properly, the data is transmitted across the USB-serial port. Two packets of information are sent to the MOTRON at different rates. The GPS solution arrives at 50 Hz and includes GPS position, velocity,

Figure 3.2: Connections of the NovAtel: A) Power Socket B) To MOTRON C) To SAR System D)&E) To IMU F) To Antenna

and ground track heading with their associated time stamps. The IMU reading arrives at 200 Hz and includes the output of the rate gyros and accelerometers with their associated time stamps.

3.2.2 MicroStrain 3DM-GX1 and Additional Functionality

While the MOTRON primarily interfaces with the NovAtel ProPak, the MOTRON is designed to be able to interface with multiple devices. There are four serial ports mounted to the back panel that can be used with additional sensors. The MicroStrain 3DM-GX1 is a smaller, lighter, and less accurate IMU that has been interfaced with the MOTRON. The 3DM-GX1 can be recorded simultaneously with the NovAtel ProPak to provide additional measurement sensors. It connects via a standard RS232 protocol and contains three rate gyros, three accelerometers, and three magnetometers to determine orientation. While the 3DM-GX1 is less accurate than the NovAtel, it may be advantageous for various applications.

Figure 3.3: The antenna connected to the NovAtel ProPak V3.

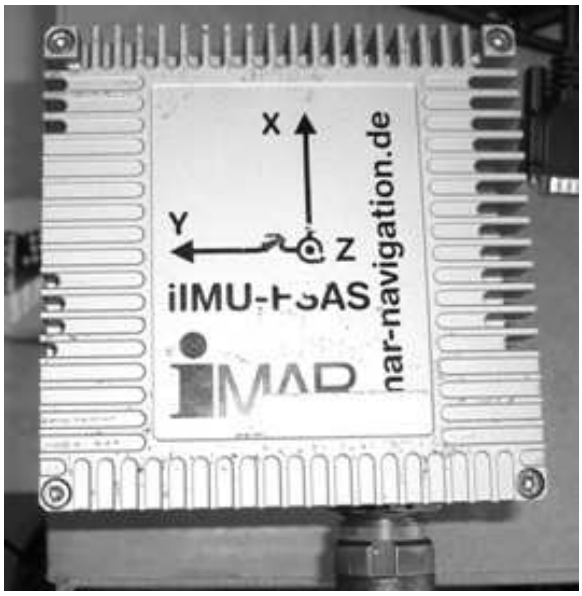

Figure 3.4: The IMAR iIMU-FSAS connected to the NovAtel ProPak V3.

The MicroStrain IMU is only one example of many different types of sensors that can be easily recorded by the MOTRON. Because there are four separate serial ports, four additional sensor interfaces can be set up simultaneously. Each additional

sensor can then be recorded to the same flash drive that all the other sensors are recorded to.

3.3 The MOTRON

In order to address each of the requirements previously stated, careful choice of components, design, and algorithms were made. The following subsections describe the components, design, user interface and algorithms of the MOTRON.

3.3.1 Components and Design

The MOTRON is based on a small ARM single board computer, the TS-7800, made by Technologic Systems. This computer utilizes a 500 Mhz ARM9 CPU, 128 MB DDR-RAM, 512 MB of NAND Flash memory, 2 USB ports, a gigabit Ethernet connection, 10 serial ports, and boots Linux (Debian) from the onboard Flash memory. The user interface consists of a small serial graphic LCD and twelve button serial keypad. The LCD screen is a blue/white 128×64 graphic display made by MicroController Pros Corporation. It is controlled through simple ASCII commands through a standard RS232 protocol. The keypad interface is a basic 3×4 keypad array made by Storm Interface. Connected to the keypad is a keypad switch that converts the keypad entries into ASCII symbols and transmits them over a serial RS232 protocol. Since the display is input only and the keypad is output only, I saved a serial port by setting the LCD and keypad to operate at the same baud rate and use the same serial port.

Each of these components are put together inside of a grated metal box shown in Fig. 3.5. The grating allows proper airflow for cooling. Each of the necessary connections to the computer are also brought to the outside of the box using the standard connections shown in Fig. 3.6. The MOTRON is mounted to a metal plate with four bolts which makes it easy to attach inside an aircraft.

This design is intuitive and easy to use. Each of the cord sockets are clearly labeled in order to avoid confusion, and the user interface is designed to be clear and understandable.

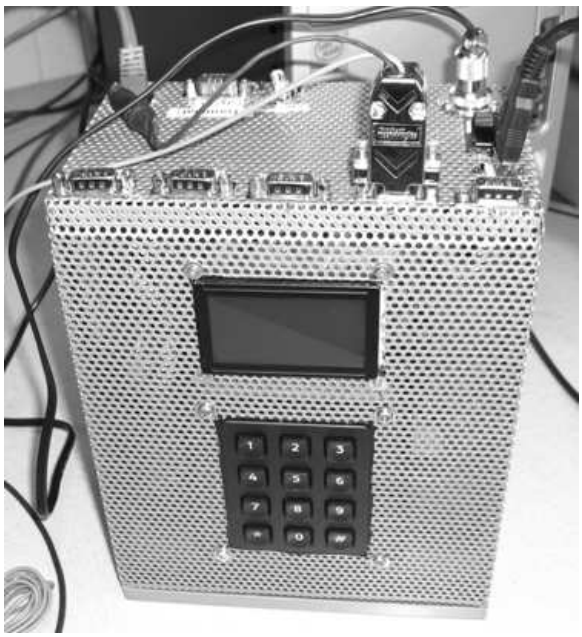

Figure 3.5: The complete MOTRON. The box size is approximately 6" × 4" × 11".

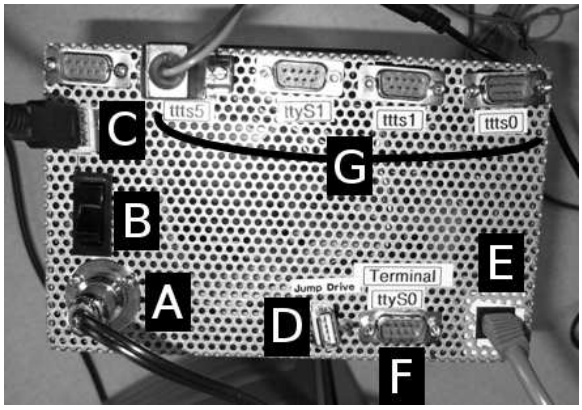

Figure 3.6: MOTRON connections A) Power Cable, B) Power Switch, C) ProPak USB Connection, D) Jump Drive Connection, E) Ethernet Connection, F) Serial Terminal Access, G) Additional Serial Ports for additional measurement instruments.

3.3.2 User Interface, Data Storage, and Input Algorithms.

In order to make the user interface as intuitive as possible, great care was taken in designing the user input system. When the MOTRON is first turned on, it displays the baud rate at which the screen is functioning (Fig. 3.7 (a)), then displays the "Data Capture" screen shown in Fig. 3.7 (b).

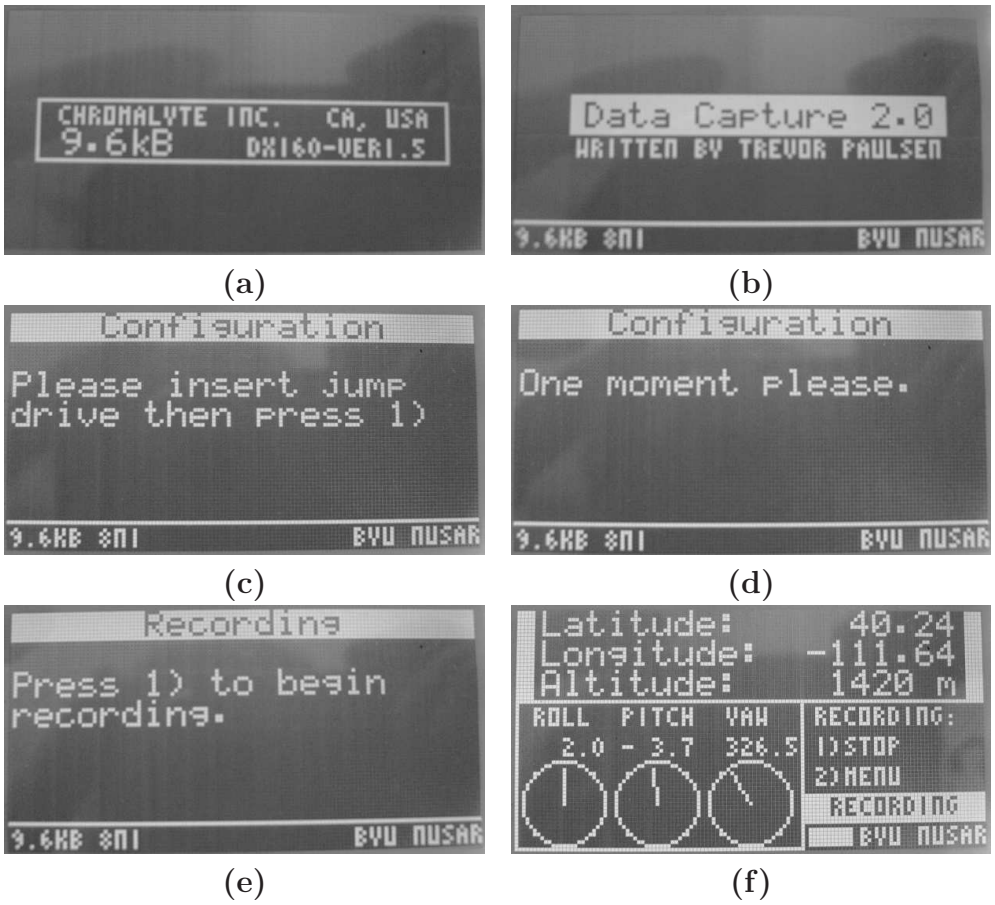

Figure 3.7: Display screens for operation of the MOTRON.

At this point, the MOTRON is ready to go and displays the following screen shown in Fig. 3.7 (c). The MOTRON determines whether or not a flash drive has been inserted. If a flash drive has not been inserted correctly, it continues to prompt the user to insert a flash drive. Once a flash drive has been inserted, the MOTRON displays the following screens shown in Figs. 3.7 (d) and (e).

Following this, the user is ready to begin collecting data. It should be noted that the NovAtel ProPak should be allowed up to two minutes to self-calibrate before attempting to record any data. Pressing one on the MOTRON keypad begins the recording process and sends a command to the NovAtel ProPak to begin sending 1 Hz time stamps to the SAR system for time syncing. If all systems are functioning correctly, the screen shows the information in Fig. 3.7 (f).

Figure 3.7 (f) shows the MOTRON's data display screen during data collection. The screen is updated in real time as the plane flies. This is performed with a multi-threaded algorithm that takes in data through the USB and displays it at a much lower rate than it is received. Simultaneously, all of the data streams are recorded to text files on the inserted flash drive. Two separate files are recorded. One file, "INS.txt", records the GPS data at 50 Hz. The second file "RAW.txt" contains the IMU data at 200 Hz. "INS.txt" includes the time stamp, latitude, longitude, altitude, north, east, and up ground velocity, ground heading, and the NovAtel solutions to roll and pitch. "RAW.txt" includes the time stamp and output of the accelerometers and gyros. The top of the IMU defines the coordinate system used by NovAtel (y should point out the nose of the plane).

In order for the MOTRON to interface correctly with the NovAtel ProPak, a new USB driver had to be cross-compiled. The new driver is a modification of the "usbserial" driver included in Linux 2.6 distributions. Using this driver, it is easy to implement a USB-serial interface with the commonly used SerialStream C++ library.

Being able to modify the MOTRON is important. To modify the MOTRON, access to the command line is necessary. There are two methods of gaining access to the MOTRON's command line in order to modify its operation. The first method is connecting serially through the "Terminal ttyS0" RS232 port at 115200 baud using the username "root" and password "hello". The second method is to ssh through the network connection with the command "ssh root@192.168.0.50" using the password "hello". Once a connection has been made, the user can change the MOTRON interface by editing the file "INS.cpp" located in the "\BYU\Interface" directory. Recompiling INS.cpp is done with the command "g++ -lserial -pthread -o INS INS.cpp". Once compiled, the program runs automatically upon bootup via an "\etc\init.d" script.

3.3.3 Interpreting the Recorded NovAtel Data

The GPS and IMU packets arrive in double precision binary format. From the GPS packets, latitude, longitude roll, pitch, and heading are given in degrees,

Table 3.1: IMU Scale Factors

Sensor	Scale Factor
Gyroscope	$\frac{\pi}{1.65888\times10^9}$ rad/s
Accelerometer	$\frac{5}{1.6384\times10^4}$ m/s^2

and altitude is given in meters. North, east, and up velocity are given in m/s. From the IMU packets, accelerometer readings are given by the change in "velocity count" along a certain axis. Similarly, the gyro measurements are given in change in "angle count". This means that the output of the IMU packet ("RAW.txt") must be scaled by a constant after it has been recorded in order to make the data meaningful. The scale factors used to scale the data are given in Table 3.1.

Multiplying the output of the IMU by these scale factors yields data that is much more useful. By doing so, we get the instantaneous angular rate in radians and instantaneous acceleration in meters per second at each sample. It is also important to change the GPS readings (found in "INS.txt") into a more useful form. First by changing ground track heading into radians by multiplying by $\pi/180$, then by converting latitude and longitude into northing and easting. This is accomplished by the MATLAB script shown in Fig. 3.8.

3.4 NovAtel Data Collected by the MOTRON

After all of the data has been recorded from the NovAtel, it must be formatted correctly so that we can begin to work with it. An example data set is shown in Figs. 3.9 and 3.10. Figure 3.9 shows the output of the accelerometers. Figure 3.10 shows the output of the gyros during the same measurement period.

3.4.1 Measurement Statistics

As can be seen in Figs. 3.9 and 3.10, the accelerometers and rate gyros are very sensitive and the signal to noise ratio is high. The statistical attributes of these six measurements plus the GPS measurements are given in Table 3.2. Because the

```
% INS(1:3,:) = Latitude, Longitude, Altitude respectively

% compute nominal segment center point
mo_window = kaiser(length(INS),2.5)';    % define weighting window
mo_window = mo_window/mean(mo_window);   % normalize window amplitude
lat0 = mean(INS(2,:).*mo_window);
lon0 = mean(INS(3,:).*mo_window);
alt0 = mean(INS(4,:).*mo_window);
time0 = mean(INS(1,:).*mo_window);

% compute positions in meters relative to center point
DTR=pi/180;                              % Degrees to radians
REsemi=6378.1363e3;                      % Semimajor axis of Earth in m
Kflat=1/298.257;                         % Earth flatness constant

% convert GPS lat/lon to displacement on locally
% tangent plane, use gps altitude as is
RE = (1-Kflat*sin(lat0*DTR).^2)*REsemi;   % local earth radius
rel_easting = RE*cos(INS(2,:)*DTR).*sin((INS(3,:)-lon0)*DTR);
rel_northing = RE*(sin((INS(2,:)-lat0)*DTR)+cos(INS(2,:)*DTR).*...
                (1-cos((INS(3,:)-lon0)*DTR))*sin(lat0*DTR));
```

Figure 3.8: A script for converting latitude and longitude to northing and easting in meters.

variance of a signal is the standard deviation squared, these measurements are useful in helping to determine the measurement noise covariance for the Kalman filter.

3.5 Conclusion

In this chapter I have discussed the need for the MOTRON, what the MOTRON is, how it was made, and what it does. I have presented the ways that the MOTRON fulfills the needs of SAR image processing, and the ways in which it was designed to make it easy to operate and utilize. A guide for operating the MOTRON was also presented as well as sample data. Useful data for determining measurement noise covariance was shown, and a method for obtaining initial conditions was explained. Overall, the MOTRON has been very successful in meeting the needs of the BYU MERS SAR team and will continue to be useful throughout the foreseeable future.

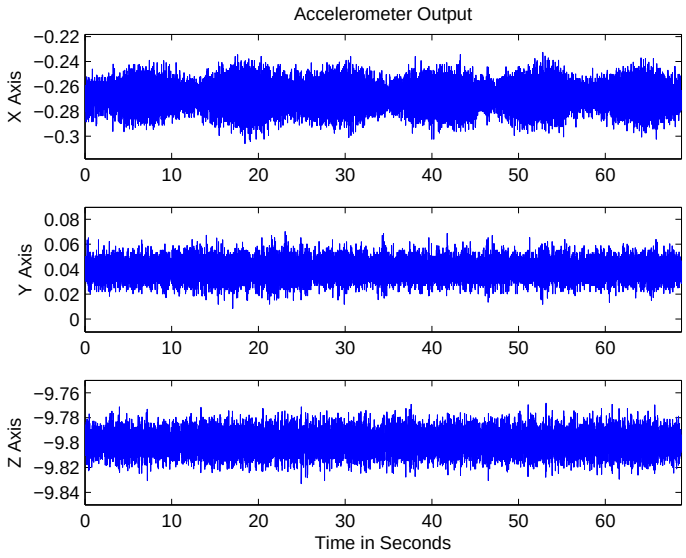

Figure 3.9: Stationary accelerometer readings over 70 seconds.

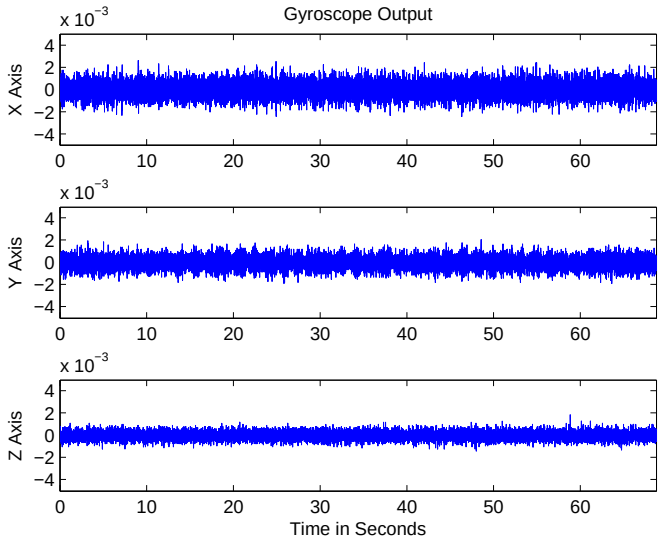

Figure 3.10: Stationary gyroscope readings over 70 seconds.

Table 3.2: Statistics of Measurements shown in Figures 3.9 and 3.10

Sensor	Mean Value	Standard Deviation
Accel X	-0.2682	0.0107
Accel Y	0.0078	0.0396
Accel Z	-9.8000	0.0091
Gyro X	-8.5849×10^{-6}	7.0660×10^{-4}
Gyro Y	-4.8611×10^{-5}	5.5001×10^{-4}
Gyro Z	-4.6365×10^{-5}	3.5382×10^{-4}
Latitude	40.2470	3.24736×10^{-6}
Longitude	-111.6480	2.1034×10^{-6}
Altitude	1.4196×10^{3}	0.4948
N Velocity	0.0017	0.0030
E Velocity	-0.0016	0.0051
U Velocity	0.0023	9.0529×10^{-4}

Chapter 4

Implementation the Square-root Unscented Kalman Filter

State estimation using the Kalman Filter is very useful for a wide range of applications. In the application for which the MOTRON is designed, imaging synthetic aperture radar (SAR) data, the attitude and velocity measurements are crucial to forming high precision radar images. Therefore, it is extremely important that the attitude, position, and velocity data are as accurate as possible. For this reason, the input sensors are some of the best sensors available. By further processing this sensor data using a Kalman filter, the accuracy and utility of the data is improved.

The following chapter outlines a method of estimating attitude and position data using the square-root unscented Kalman filter presented in Chapter 2. Section 4.1 summarizes the dynamics of aircraft flight and propose a set of state variables for modeling these dynamics. Section 4.2 defines a model for the output sensors of the system. Finally, Section 4.3 presents the details of setting up an unscented Kalman filter to implement the state space model to be presented.

4.1 Aircraft Flight Dynamics and State Variables

Prior to developing the filter, it is important to first look at the dynamics of the system to be modeled. By understanding the underlying principles of aircraft flight, we can more accurately model it. The expressions presented are general to any rigid body and are derived from equations presented in Chapter 2.

4.1.1 State Variables

The state variables of the aircraft are the following eighteen quantities:

$$
\begin{aligned}
p_n &= \text{Northing position of plane,} \\
p_e &= \text{Easting position of plane,} \\
p_d &= \text{Down position of plane (negative altitude),} \\
u &= \text{Body frame velocity in the } \hat{i} \text{ direction,} \\
v &= \text{Body frame velocity in the } \hat{j} \text{ direction,} \\
w &= \text{Body frame velocity in the } \hat{k} \text{ direction,} \\
ai_{nc} &= \text{Non-centripetal acceleration in the } \hat{i} \text{ direction,} \\
aj_{nc} &= \text{Non-centripetal acceleration in the } \hat{j} \text{ direction,} \\
ak_{nc} &= \text{Non-centripetal acceleration in the } \hat{k} \text{ direction,} \\
\phi &= \text{Roll angle,} \\
\theta &= \text{Pitch angle,} \\
\psi &= \text{Heading angle,} \\
p &= \text{Angular rate along } \hat{i}, \\
q &= \text{Angular rate along } \hat{j}, \\
r &= \text{Angular rate along } \hat{k}, \\
B_i &= \hat{i} \text{ gyro bias,} \\
B_j &= \hat{j} \text{ gyro bias,} \\
B_k &= \hat{k} \text{ gyro bias.}
\end{aligned} \tag{4.1}
$$

As shown in Fig. 4.1, the $\hat{i}$, $\hat{j}$, $\hat{k}$ directions are in the body frame of the aircraft. This means that $\hat{i}$ is in the direction of the nose of the plane, $\hat{j}$ is pointing out the right wing, and $\hat{k}$ points out the bottom of the aircraft forming a right hand coordinate system. It is also important to note that the angular rates (p, q, r) as well as the Euler angles (ϕ, θ, ψ) follow the right hand rule. The position of the aircraft, p_n, p_e, and p_d, are given in the world frame, $\hat{x}$, $\hat{y}$, and $\hat{z}$, which correspond to northing,

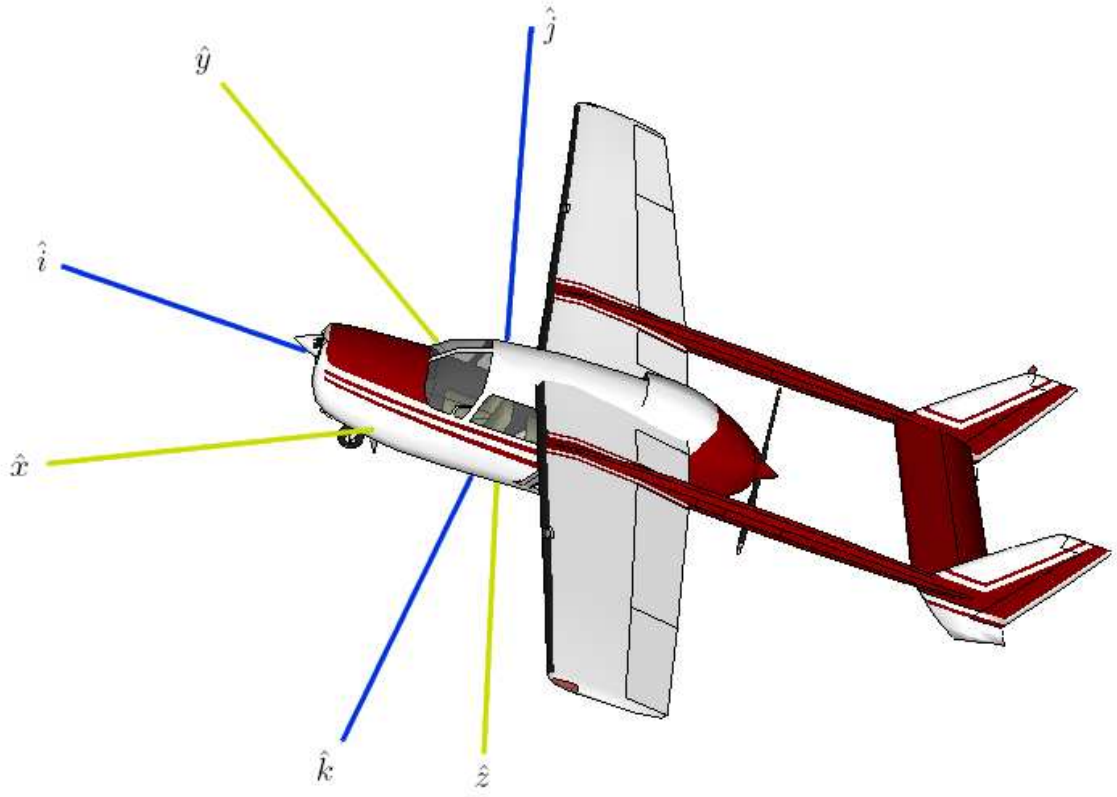

Figure 4.1: Body frame versus world frame. $\hat{i}$, $\hat{j}$, and $\hat{k}$ represent the three orthogonal directions of the aircraft's body frame. $\hat{x}$, $\hat{y}$, and $\hat{z}$ represent the three orthogonal directions in the world frame.

easting, and negative altitude provided by GPS measurements respectively. The body frame velocity (u, v, w), angular rates (p, q, r), non-centripetal acceleration (ai_{nc}, aj_{nc}, ak_{nc}), and gyro biases (B_i, B_j, B_k) are represented in the body frame, ($\hat{i}$, $\hat{j}$, $\hat{k}$).

4.1.2 Reference Frames

The body frame of the aircraft is related to the world frame($\hat{x}$, $\hat{y}$, $\hat{z}$) through the following rotation matrix:

$$R_{BtoW} = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}. \tag{4.2}$$

This matrix takes vectors within the body frame ($\hat{i}$, $\hat{j}$, $\hat{k}$) and rotates them into the world frame ($\hat{x}$, $\hat{y}$, $\hat{z}$). In order to move from the world frame to the body frame, we simply multiply by R_{WtoB} which is the transpose of R_{BtoW}.

$$R_{WtoB} = R_{BtoW}^{T}. \tag{4.3}$$

4.1.3 Equations of Motion

In order to calculate the northing, easting, and down velocities ($\dot{p}_n$, $\dot{p}_e$, $\dot{p}_d$), we rotate the body frame velocity (u, v, w) through R_{BtoW}. This gives us the velocity in the world frame ($\dot{p}_n$, $\dot{p}_e$, and $\dot{p}_d$)

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = R_{BtoW} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{4.4}$$

yielding

$$\begin{aligned} \dot{p}_n &= u(c\theta c\psi) + v(s\phi s\theta c\psi - c\phi s\psi) + w(c\phi s\theta c\psi + s\phi s\psi), \\ \dot{p}_e &= u(c\theta s\psi) + v(s\phi s\theta s\psi + c\phi s\psi) + w(c\phi s\theta s\psi - s\phi c\psi), \\ \dot{p}_d &= u(-s\theta) + v(s\phi c\theta) + w(c\phi c\theta). \end{aligned} \tag{4.5}$$

The body frame acceleration, ($\dot{u}$, $\dot{v}$, $\dot{w}$), can be modeled as the sum of the non-centripetal acceleration and centripetal acceleration in the body frame. Centripetal acceleration is found by taking the cross product of the angular rates with the body frame velocity. This is known as the Coriolis equation,

$$\begin{bmatrix} ai_c \\ aj_c \\ ak_c \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{4.6}$$

and yields

$$\begin{aligned} ai_c &= qw - rv, \\ aj_c &= ru - pw, \\ ak_c &= pv - qu. \end{aligned} \tag{4.7}$$

Eq. 4.7 gives an expression for centripetal acceleration which can be used to find the total body frame acceleration,

$$\begin{aligned} \dot{u} &= ai_{nc} + ai_c = ai_{nc} + qw - rv, \\ \dot{v} &= aj_{nc} + aj_c = aj_{nc} + ru - pw, \\ \dot{w} &= ak_{nc} + ak_c = ak_{nc} + pv - qu. \end{aligned} \tag{4.8}$$

Eq. 4.8 gives a representation of acceleration in the aircraft's body frame. The Euler angle rates can be described using p, q, and r through the following rotation as shown by [4],

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{4.9}$$

Finally, this rotation takes each of the three angular rates in the body frame, and translates them into changes in the Euler angles within the world frame. After multiplying, the following is found:

$$\begin{aligned} \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta, \\ \dot{\theta} &= q\cos\phi - r\sin\phi, \\ \dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta. \end{aligned} \tag{4.10}$$

Finding dynamic equations of ai_{nc}, aj_{nc}, and ak_{nc}, also referred to as "jerk", can be difficult. Furthermore, modeling the gyro biases B_i, B_j, and B_k is also trou-

blesome. For the sake of this problem we assume that

$$\dot{\boldsymbol{a}_{nc}} = -0.1\boldsymbol{a}_{nc},$$
$$\dot{\boldsymbol{\Omega}} = 0,$$
and
$$\dot{\boldsymbol{B}} = 0. \tag{4.11}$$

The assumption behind the acceleration model is that if the jerk is non-zero, the jerk is expected to decrease. We also assume that the gyro bias does not change, and model $\dot{\boldsymbol{\Omega}}$ as having no change. These assumptions have given reasonable results. Using Eqs. 4.5, 4.8, 4.10, and 4.11, the dynamics of the aircraft are formed through the following set of state space equations:

$$\begin{aligned}
\dot{p}_n &= u(c\theta c\psi) + v(s\phi s\theta c\psi - c\phi s\psi) + w(c\phi s\theta c\psi + s\phi s\psi),\\
\dot{p}_e &= u(c\theta s\psi) + v(s\phi s\theta s\psi + c\phi s\psi) + w(c\phi s\theta s\psi - s\phi c\psi,\\
\dot{p}_d &= u(-s\theta) + v(s\phi c\theta) + w(c\phi c\theta),\\
\dot{u} &= ai_{nc} + qw - rv,\\
\dot{v} &= aj_{nc} + ru - pw,\\
\dot{w} &= ak_{nc} + pv - qu,\\
\dot{a}i_{nc} &= -0.1ai_{nc},\\
\dot{a}j_{nc} &= -0.1aj_{nc},\\
\dot{a}k_{nc} &= -0.1ak_{nc},\\
\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta,\\
\dot{\theta} &= q\cos\phi - r\sin\phi,\\
\dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta,\\
\dot{p} &= 0,\\
\dot{q} &= 0,\\
\dot{r} &= 0,
\end{aligned}$$

$$\dot{B}_i = 0,$$
$$\dot{B}_j = 0,$$
$$\dot{B}_k = 0. \tag{4.12}$$

Eq. 4.12 defines the dynamic nonlinear function $f(\boldsymbol{x}, t)$ of our state space.

4.2 Measurement Sensors

The system input includes accelerometers, fiber-optic rate gyros, and high precision GPS measurements. The output of a rate gyro is modeled by

$$y_{gyro} = \Omega + B_{gyro} + \eta_{gyro}, \tag{4.13}$$

where y_{gyro} is the output of the gyro, Ω is the angular rate, B_{gyro} is the gyro bias, and η is zero mean white noise. The output of an accelerometer is modeled as

$$y_{accel} = a_{nc} + a_c + (R_{WtoB})g_{lift} + \eta_{accel}, \tag{4.14}$$

where y_{accel} is the output of the accelerometer, a_{nc} is non-centripetal acceleration, a_c is centripetal acceleration, g_{lift} is the lift acceleration against gravity (-9.8 m/s in the $\hat{z}$ direction) rotated from the world to body frame, and η_{accel} is zero mean white noise.

The measurements of position p_n, p_e, and p_d are directly measured by the GPS system, as are $\dot{p}_n$, $\dot{p}_e$, and $\dot{p}_d$, the velocity in the world frame. Putting these equations together, the output of the accelerometers can be modeled using the following

$$yi_{accel} = ai_{nc} + (qw - rv) + g_{lift}(-\sin\theta),$$
$$yj_{accel} = aj_{nc} + (ru - pw) + g_{lift}(\cos\theta\sin\phi),$$
$$yk_{accel} = ak_{nc} + (pv - qu) + g_{lift}(\cos\theta\cos\phi). \tag{4.15}$$

The gyros can be modeled as

$$\begin{aligned} yi_{gyro} &= p + B_i, \\ yj_{gyro} &= q + B_j, \\ yk_{gyro} &= r + B_k, \end{aligned} \tag{4.16}$$

and the GPS position can be modeled

$$\begin{aligned} \text{Northing} &= p_n, \\ \text{Easting} &= p_e, \\ \text{Altitude} &= -p_d, \end{aligned} \tag{4.17}$$

and lastly, the world frame velocity is given by

$$\begin{aligned} \dot{p}_n &= u(c\theta c\psi) + v(s\phi s\theta c\psi - c\phi s\psi) + w(c\phi s\theta c\psi + s\phi s\psi), \\ \dot{p}_e &= u(c\theta s\psi) + v(s\phi s\theta s\psi + c\phi s\psi) + w(c\phi s\theta s\psi - s\phi c\psi), \\ \dot{p}_d &= u(-s\theta) + v(s\phi c\theta) + w(c\phi c\theta)(c\phi s\theta c\psi + s\phi s\psi). \end{aligned} \tag{4.18}$$

Each of these equations define an output function for the Kalman filter.

4.3 Implementing a Square-root Unscented Kalman Filter

With these set of equations, we can define $f(\boldsymbol{x}_t, t)$ and $h(\boldsymbol{x}_t, t)$, the nonlinear functions of a state space model. General implementation of the square-root unscented Kalman Filter (SR-UKF) is outlined in Chapter 2. This section discusses a particular implementation of the SR-UKF in order to solve the state space model

$$\boldsymbol{x}_{i|i-1} = f(\boldsymbol{x}_{i-1|i-1}, t) + \xi, \tag{4.19}$$

$$\boldsymbol{y} = h(\boldsymbol{x}_{i|i-1}, t) + \eta, \tag{4.20}$$

where ξ and η represent process and measurement noise respectively. The SR-UKF uses a set of scalar weights,

$$\begin{aligned} W_0^m &= \lambda/(L+\lambda), \\ W_0^c &= \lambda/(L+\lambda) + (1-\alpha^2+\beta), \\ W_i^m &= W_i^c = 1/[2(L+\lambda)]), \end{aligned} \tag{4.21}$$

where

$$\lambda = L(\alpha^2 - 1), \tag{4.22}$$

is a scaling parameter. Values of $\alpha = 1.5$, $\beta = 2$ and $L = 18$ are used throughout the SR-UKF implementation. α determines the spacing of the sigma points, β should be equal to two for Gaussian noise, and L represents the number of states in the system.

4.3.1 Finding Initial Conditions

In order to initialize the Kalman filter, the MOTRON's data set must provide initial conditions for each of the states in the Kalman filter. In order to simplify this process, the IMU is allowed to sit still for a few minutes while initial conditions are calibrated. By allowing a calibration time, we can assume the initial position is the mean position of that period. The initial velocity is set to zero and the initial attitude is calculated in a straightforward method.

The lift force against gravity is assumed to be -9.8 m/s^2 in the $\hat{z}$ direction. Using the rotation matrix $\boldsymbol{R}_{WtoB}$ from Eq. 4.2, we can determine the initial roll and pitch.

$$\begin{bmatrix} \bar{A}_X \\ \bar{A}_Y \\ \bar{A}_Z \end{bmatrix} = \boldsymbol{R}_{WtoB} \begin{bmatrix} 0 \\ 0 \\ -9.8 \end{bmatrix}, \tag{4.23}$$

giving

$$\begin{aligned}\bar{A}_X &= -\sin\theta(-9.8),\\ \bar{A}_Y &= \sin\phi\cos\theta(-9.8),\\ \bar{A}_Z &= \cos\phi\cos\theta(-9.8),\end{aligned} \tag{4.24}$$

which leads to

$$\theta_0 = \sin^{-1}\frac{\bar{A}_X}{9.8}, \tag{4.25}$$

$$\phi_0 = \cos^{-1}\frac{\bar{A}_Z}{-9.8\cos\theta}, \tag{4.26}$$

where $\bar{A}$ represents the mean value of the acceleration over the calibration time. θ_0 represents the initial pitch and ϕ_0 represents the initial roll. After calculating the initial pitch, it is plugged into Eq. 4.26. Finding the initial heading is done in a similar manner to Eqs. 4.25 and 4.26 but using the gyros,

$$\begin{bmatrix}\bar{G}_X\\ \bar{G}_Y\\ \bar{G}_Z\end{bmatrix} = R_{WtoB}\begin{bmatrix}\Omega\cos(\bar{\zeta})\\ 0\\ \Omega\sin(\bar{\zeta})\end{bmatrix}. \tag{4.27}$$

$\bar{G}$ represents the average gyroscope reading during the calibration period, Ω represents the rotation of the Earth (7.2722×10^{-5} rad/s), and $\bar{\zeta}$ represents the average latitude over the calibration period. This equation yields

$$\bar{G}_X = \Omega\cos\bar{\zeta}(\cos\theta\cos\psi) + \Omega\sin\bar{\zeta}(-\sin\theta), \tag{4.28}$$

giving

$$\psi_0 = \cos^{-1}\frac{\bar{G}_X + \Omega\sin\bar{\zeta}\sin\theta}{\Omega\cos\bar{\zeta}\cos\theta}. \tag{4.29}$$

Therefore, by solving for θ, then ϕ, then ψ using Eqs. 4.25, 4.26, and 4.29, we have a method for obtaining the initial values of pitch, roll, and heading. Initial velocities

are then set to zero. This approach gives a consistent method for obtaining initial conditions.

4.3.2 Time Update

After computing initial conditions, the first step is the sigma point calculation and time update shown by the equations below:

$$\boldsymbol{\chi_{i-1}} = \left[\begin{array}{ccc} \hat{\boldsymbol{x}}_{\boldsymbol{i-1}} & \hat{\boldsymbol{x}}_{\boldsymbol{i-1}} + \eta \boldsymbol{S}_i & \hat{\boldsymbol{x}}_{\boldsymbol{i-1}} - \eta \boldsymbol{S}_i \end{array} \right], \tag{4.30}$$

$$\boldsymbol{\chi_{i|i-1}} = f(\boldsymbol{\chi_{i-1}}), \tag{4.31}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{i|i-1}} = f(\hat{\boldsymbol{x}}_{\boldsymbol{i-1}}), \tag{4.32}$$

$$\boldsymbol{S}_{i|i-1} = \mathrm{qr} \left[\begin{array}{cc} \sqrt{W_i^c}(\boldsymbol{\chi}_{i|i-1}(1:2L) - \hat{\boldsymbol{x}}_{\boldsymbol{i|i-1}}) & \sqrt{Q} \end{array} \right], \tag{4.33}$$

$$\boldsymbol{S}_{i|i-1} = \mathrm{cholupdate} \left[\begin{array}{ccc} \boldsymbol{S}_{i|i-1}, & \boldsymbol{\chi_{i|i-1}}(0), & \mathrm{sign}\{W_0^c\} \end{array} \right]. \tag{4.34}$$

As shown in Eq. 4.30, the SRUKF relies on a deterministic "sampling" approach to calculate mean and covariance terms. This equation determines how those points are chosen. $\boldsymbol{\chi_{i-1}}$ is an $L \times 2L+1$ or 18×37 matrix where each column represents a sigma point. In Eq. 4.31, each of these columns or points are passed through the nonlinear function f and stored in $\boldsymbol{\chi}_{i|i-1}$. Typically the mean value, $\hat{\boldsymbol{x}}_i$ is computed using a weighted average of the sigma points, however, in this implementation the new mean value is computed by simply passing the previous sample mean $\hat{\boldsymbol{x}}_{i-1}$ through f, shown in Eq. 4.32. The new Cholesky factor of the state covariance, $\boldsymbol{S}_i$, is calculated in two steps, shown in Eqs. 4.33 and 4.34. First, the Cholesky factor, $\boldsymbol{S}_i$, is calculated using a QR decomposition of the compound matrix square-root of the additive process noise covariance. The subsequent Cholesky update (or downdate) in Eq. 4.34 is necessary since the zero'th weight, W_0^c, can be negative with different values of α. Fig. 4.2 illustrates a MATLAB script to accomplish the time update portion of the SR-UKF.

```
function [new_xhat new_Si] = timeupdate(old_xhat, old_Si)

    L = number_of_states;
    T = sample_time;
    old_sigma_points=[old_xhat old_xhat(:,ones(1, L))+eta*old_Si...
                                old_xhat(:,ones(1, L))-eta*old_Si];
    new_sigma_points = f(old_sigma_points);
    new_xhat = new_sigma_points(:,1);
    [Junk new_Si] = qr([Wci*(new_sigma_points(:,2:end)-...
                        new_xhat(:,ones(1,L*2))) chol(Q*T)].');
    new_Si = new_Si(1:L,:);
    if Wc1 > 0
        Sip = cholupdate(new_Si, Wc1*(new_sigma_points(:,1) -...
                                      new_xhat), '+');
        p = 0;
    else
        [Sip,p] = cholupdate(new_Si, Wc1*(new_sigma_points(:,1)...
                             - new_xhat), '-');
    end
    if p > 0,
        fprintf('new_Si is negative definite.');
    else
        new_Si=Sip;
    end
```

Figure 4.2: Time update MATLAB script.

4.3.3 Measurement Update

The measurement update equations follow a similar pattern.

$$\hat{\boldsymbol{y}}_{i|i-1} = h(\boldsymbol{x}_{i|i-1}), \tag{4.35}$$

$$\boldsymbol{\mathcal{Y}_{i|i-1}} = h(\boldsymbol{\chi}_{i|i-1}), \tag{4.36}$$

$$\boldsymbol{S}_{y_i} = \mathrm{qr}\left[\ \sqrt{W_i^c}(\boldsymbol{\mathcal{Y}_{i|i-1}}(1:2L) - \hat{\boldsymbol{y}}_{i|i-1}) \quad \sqrt{\boldsymbol{R}}\ \right], \tag{4.37}$$

$$\boldsymbol{S}_{y_i} = \mathrm{cholupdate}\left[\ \boldsymbol{S}_{y_i},\ \ \boldsymbol{\mathcal{Y}_{i|i-1}}(0),\ \ \mathrm{sign}\{W_0^c\}\ \right], \tag{4.38}$$

$$\boldsymbol{P}_{x_i y_i} = \sum_{k=0}^{2L} W_k^c \left[\ \boldsymbol{\chi}_{i|i-1}(k) - \hat{\boldsymbol{x}}_i\ \right] \left[\ \boldsymbol{\mathcal{Y}}_{i|i-1}(k) - \hat{\boldsymbol{y}}_{i|i-1}\ \right]^T, \tag{4.39}$$

$$\boldsymbol{\mathcal{K}_i} = (\boldsymbol{P}_{x_i y_i} / \boldsymbol{S}_{y_i}^T) / \boldsymbol{S}_{y_i}, \tag{4.40}$$

$$\hat{\boldsymbol{x}}_{i|i} = \hat{\boldsymbol{x}}_{i|i-1} + \boldsymbol{\mathcal{K}}_i (\boldsymbol{y}_{measured} - \hat{\boldsymbol{y}}_{i|i-1}), \tag{4.41}$$

$$\boldsymbol{U} = \boldsymbol{\mathcal{K}}_i \boldsymbol{S}_{yi}, \tag{4.42}$$

$$\boldsymbol{S}_{i|i} = \text{cholupdate} \left[\begin{array}{ccc} \boldsymbol{S}_{i|i-1}, & \boldsymbol{U}, & \text{`-'} \end{array} \right]. \tag{4.43}$$

The output estimate $\hat{\boldsymbol{y}}_{i|i-1}$ is computed by passing $\hat{\boldsymbol{x}}_{i|i-1}$ through the nonlinear function h. Next, a set of output points $\boldsymbol{\mathcal{Y}_{i|i-1}}$ is calculated by passing each of the new sigma points, $\boldsymbol{\chi_{i|i-1}}$, through the nonlinear output function h. A similar two-step approach is applied to the calculation of the Cholesky factor, $\boldsymbol{S}_{yi}$, of the observation-error covariance in Equations 4.37 and 4.38, where R is the measurement noise covariance. Two nested inverse (or least squares) solutions are used to calculate the Kalman gain, $\boldsymbol{\mathcal{K}}_i$, in Eq. 4.40. Finally, the posterior measurement update of the Cholesky factor of the state covariance is calculated in Eq. 4.43. A MATLAB script used to calculate the measurement update is included in Fig. 4.3

4.3.4 Nonlinear State Functions

Now that we have sufficiently set up the Kalman filter to optimize our measurements, we introduce the functions that the filter operates with. Using Eq. 4.12, we can construct f. If we represent Eq. 4.12 as the time derivative of the state $\hat{\boldsymbol{x}}$ then,

$$f(\hat{\boldsymbol{x}}_{i-1}) = \hat{\boldsymbol{x}}_{i-1} + \Delta \dot{\hat{\boldsymbol{x}}}_{i-1}, \tag{4.44}$$

where Δ is the change in time, or sample rate. This is an approximation of the nonlinear function f. If Δ is sufficiently small, then the approximation good. The function f must be made to accept large arrays of points, so in the following MATLAB code the '.*' operator is used in place of '*'. The MATLAB script can be seen in Fig. 4.5. In this particular implementation we receive measurements in two different packets. We receive the IMU measurements (accelerometers and gyros) at a rate of 200 Hz and we receive GPS measurements (position and ground speed) at a rate of

```
function [new_xhat new_Si] = measureupdate(xhat, Si,...
                                           number_of_measurements)

    M = number_of_measurements;
    sigma_points = [xhat xhat(:,ones(1, L))+eta*Si...
                         xhat(:,ones(1, L))-eta*Si];
    y_points = h(sigma_points);
    yhat = y_points(:,1);
    [Junk Syi] = qr([Wci*(y_points(:,2:end) -...
                     yhat(:,ones(1, M-1))) chol(R)].');
    if Wc1 > 0
        Syi = cholupdate(Syi(1:M,:), Wc1*(y_points(:,1)...
                                          - yhat), '+');
    else
        Syi = cholupdate(Syi(1:M,:), Wc1*(y_points(:,1)...
                                          - yhat), '-');
    end
    Syi = Syi.';
    Pxy = Wc1*(sigma_points(:,1) - xhat)*(y_points(:,1) - yhat).';
    for j=2:L*2+1
        Pxy = Pxy + Wci*(sigma_points(:,j) - xhat)*...
                        (y_points(:,j) - yhat).';
    end
    K = (Pxy/Syi.')/Syi;
    new_xhat = xhat + K*(measurements - yhat);
    U = K*Syi;
    for j=1:M
        [Sip,p] = cholupdate(Si, U(:,j), '-');
        if p > 0,
            fprintf('Si negative definite');
        else
            new_Si=Sip;
        end
    end
```

Figure 4.3: Measurement update MATLAB script.

50 Hz. This necessitates the use of two separate h functions. The nonlinear function h_{GPS} makes use of Equations 4.17 and 4.18, and is illustrated in Fig. 4.4.

```
function [y_vel_points] = h_GPS(points)

    pn    = points(1,:);
    pe    = points(2,:);
    pd    = points(3,:);
    u     = points(4,:);
    v     = points(5,:);
    w     = points(6,:);
    phi   = points(10,:);
    theta = points(11,:);
    psi   = points(12,:);

    cphi   = cos(phi);
    sphi   = sin(phi);
    ctheta = cos(theta);
    stheta = sin(theta);
    cpsi   = cos(psi);
    spsi   = sin(psi);

    pndot = ctheta.*cpsi.*u + (sphi.*stheta.*cpsi - cphi.*spsi).*v...
            + (cphi.*stheta.*cpsi + sphi.*spsi).*w;
    pedot = ctheta.*spsi.*u + (sphi.*stheta.*spsi + cphi.*cpsi).*v...
            + (cphi.*stheta.*spsi - sphi.*cpsi).*w;
    hdot  = (-stheta.*u + sphi.*ctheta.*v + cphi.*ctheta.*w);

    y_vel_points = [pn; pe; pd; pndot; pedot; hdot; psi];
```

Figure 4.4: Nonlinear function h_{GPS} MATLAB script.

```
function [sigma_star] = f(points)

    pn    = points(1,:);
    pe    = points(2,:);
    h     = points(3,:);
    u     = points(4,:);
    v     = points(5,:);
    w     = points(6,:);
    ncax  = points(7,:);
    ncay  = points(8,:);
    ncaz  = points(9,:);
    phi   = points(10,:);
    theta = points(11,:);
    psi   = points(12,:);
    p     = points(13,:);
    q     = points(14,:);
    r     = points(15,:);
    Bi    = points(16,:);
    Bj    = points(17,:);
    Bk    = points(18,:);

    cphi = cos(phi);
    sphi = sin(phi);
    ctheta = cos(theta);
    stheta = sin(theta);
    ttheta = tan(theta);
    sectheta = sec(theta);
    cpsi = cos(psi);
    spsi = sin(psi);

    pndot = ctheta.*cpsi.*u + (sphi.*stheta.*cpsi - cphi.*spsi).*v...
            + (cphi.*stheta.*cpsi + sphi.*spsi).*w;
    pedot = ctheta.*spsi.*u + (sphi.*stheta.*spsi + cphi.*cpsi).*v...
            + (cphi.*stheta.*spsi - sphi.*cpsi).*w;
    hdot  = (-stheta.*u + sphi.*ctheta.*v + cphi.*ctheta.*w);

    udot = ncax + (q.*w - r.*v);
    vdot = ncay + (r.*u - p.*w);
    wdot = ncaz + (p.*v - q.*u);

    phidot = p + sphi.*ttheta.*q + cphi.*ttheta.*r;
    thetadot = cphi.*q - sphi.*r;
    psidot = sphi.*sectheta.*q + cphi.*sectheta.*r;

    sigma_star = [pn+T*pndot; pe+T*pedot; h+T*hdot; u+T*udot;...
                  v+T*vdot; w+T*wdot; ncax-T*0.1*ncax;...
                  ncay-T*0.1*ncay; ncaz-T*0.1*ncaz;...
                  phi+T*phidot; theta+T*thetadot; psi+T*psidot;...
                  p; q; r; Bi; Bj; Bk];
```

Figure 4.5: Nonlinear function f MATLAB script.

4.3.5 Simulated Measurements

In order to use additional information about the aircraft's dynamics, a few synthetic "measurements" are used to further correct the output of the Kalman filter. These synthetic measurements take advantage of the following formulas:

$$\begin{aligned} v_{total} &= \sqrt{u^2 + v^2 + w^2}, \\ \dot{\psi} &= \frac{g}{v_{total}} \tan \phi, \end{aligned} \tag{4.45}$$

where g is the lift acceleration against gravity, -9.8 m/s^2. Equation 4.45 means that an aircraft cannot turn unless it rolls to the right or left. This is called a coordinated turn.

The second synthetic measurement is to correct the body frame velocity. This synthetic measurement is probably the most inaccurate, so special care should be taken when tuning the measurement noise covariance matrix. The variance of these three should be tuned sufficiently high so as to not provide false accuracy to the overall output. It is given by the equations

$$\begin{aligned} u &= v_{total} \cos \alpha, \\ v &= 0, \\ w &= v_{total} \sin \alpha, \end{aligned} \tag{4.46}$$

where α is the angle of attack. For larger aircraft, α is relatively small ($\approx 0^\circ - 5^\circ$), but for some small aircraft α is larger ($\approx 5^\circ - 10^\circ$).

Finally, a third simulated measurement is meant to keep the non-centripetal acceleration in check. The simulated measurement simply says that the non-centripetal acceleration is zero. With this insight and Eqs. 4.15 and 4.16, we can construct h_{IMU}, the second h output function. Figure 4.6 shows the MATLAB script to accomplish this.

```
function [y_IMU_points] = h_IMU(points)

    g = -9.8;

    u     = points(4,:);
    v     = points(5,:);
    w     = points(6,:);
    ncax  = points(7,:);
    ncay  = points(8,:);
    ncaz  = points(9,:);
    phi   = points(10,:);
    theta = points(11,:);
    p     = points(13,:);
    q     = points(14,:);
    r     = points(15,:);
    Bi    = points(16,:);
    Bj    = points(17,:);
    Bk    = points(18,:);

    stheta = sin(theta);
    ctheta = cos(theta);
    sphi = sin(phi);
    cphi = cos(phi);

    y_acceli = ncax + (q.*w - r.*v) + g*(-stheta);
    y_accelj = ncay + (r.*u - p.*w) + g*(ctheta.*sphi);
    y_accelk = ncaz + (p.*v - q.*u) + g*(ctheta.*cphi);

    y_gyroi = p + Bi;
    y_gyroj = q + Bj;
    y_gyrok = r + Bk;

    vector = sqrt(u.^2 + v.^2 + w.^2);
    psidot = g./vector.*tan(phi);

    y_IMU_points = [y_acceli; y_accelj; y_accelk; y_gyroi;...
                    y_gyroj; y_gyrok; u; v; w; psidot; 0; 0; 0];
```

Figure 4.6: Nonlinear function h_{IMU} MATLAB script.

4.4 Conclusion

This chapter has presented a method for modeling aircraft flight dynamics and a method for applying the SRUKF for optimizing attitude and position data. I have proposed a set of state variables that can be used to model aircraft dynamics, derived a set of equations to define the motion of the aircraft, and proposed a Kalman filter

to optimize gyroscope, accelerometer, and GPS readings in synchronization with our model.

Chapter 5

Results

Once an actual flight has been recorded by the MOTRON, the SR-UKF can be applied to the collected sensor data. By applying the SR-UKF to the sensor data, a completely new data set is created that contains estimates of the aircraft's position, speed, and attitude which can be compared to the NovAtel estimates. This chapter presents the results of the SR-UKF implementation discussed in Chapter 4. First, the Kalman filter is applied to a simulated data set that contains absolute truth data. Then, the Kalman filter is applied to actual recorded data and an explanation of the results is provided.

5.1 Flight Simulator

In order to gauge the effectiveness of the SR-UKF, I first use simulated sensor and position data from a flight simulator. The flight simulator provides computer-generated accelerometer, gyroscope and GPS readings and also provides absolute truth data about the position, velocity, and attitude. By applying the SR-UKF to the computer-generated sensor data and comparing its output to the truth, it is easy to judge the effectiveness of the SR-UKF. A flight simulator created by BYU's MAGICC Lab is used to create a set of synthetic data [7]. The simulator is typically used to model small unmanned aerial vehicles. Modifications have been made to more closely simulate the sensors that are used onboard the MERS aircraft. It is constructed in MATLAB's Simulink and consists of an autopilot block and a flight dynamics block shown in Fig. 5.1. The autopilot system is outside the scope of this thesis; however, the flight dynamics block is important to the results, and is shown in Fig. 5.2.

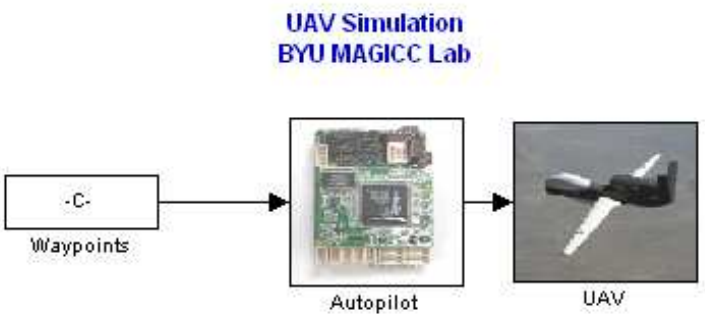

Figure 5.1: The basic Simulink model for the MAGICC Lab flight simulator consisting of an autopilot block and a flight dynamics block.

As is shown in Fig. 5.2, the output of the dynamics are fed into the simulated sensors block which generates a file "yout.mat". This file contains the data from the virtual accelerometers, gyroscopes, and GPS. Special care is taken to make the virtual sensors imitate the real sensors onboard the MERS aircraft by matching the variance of the simulated noise to that of the measured noise given in Chapter 3. The file "xout.mat" contains all of the truth data from the dynamics block related to the Kalman filter states. The flight dynamics block determines how the simulated aircraft responds to external stimulus. It relies on a complex set of coefficients that model an aircraft's drag, lift, propeller thrust, weight, aerodynamics, and inertia, [8] [9]. Every aircraft has a distinct and unique set of coefficients and responds to autopilot commands differently. For this reason, the autopilot must be specifically tuned for each individual aircraft.

All of the noise parameters, flight dynamics coefficients, and simulation parameters are given in an initialization file, "param.m", which can be readily modified. Additionally, within the "sensors" block are the sensor models. The sensor block takes the true position, velocity, forces, angular rates, and attitude of the aircraft and cre-

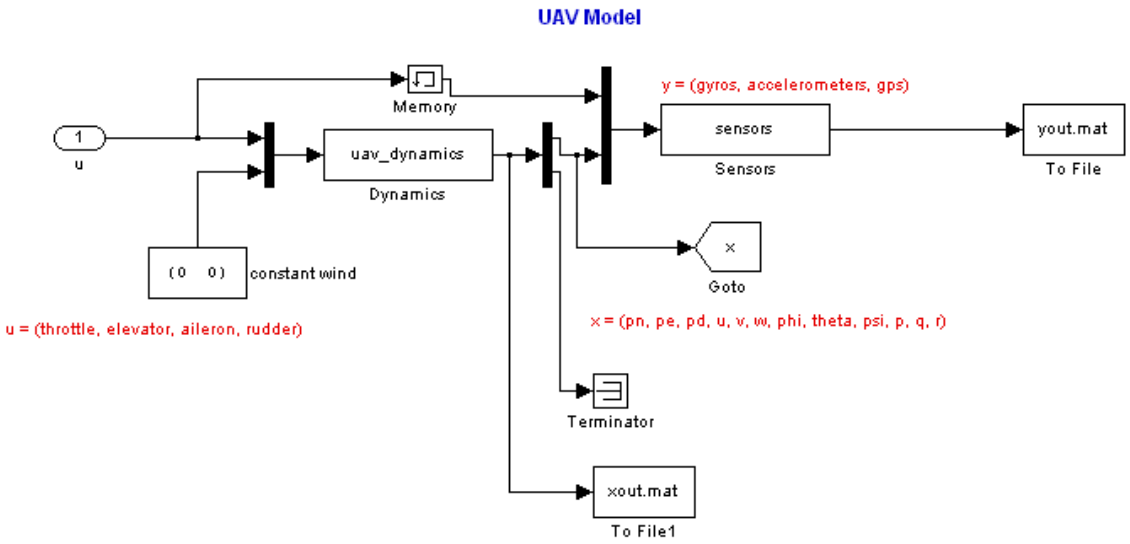

Figure 5.2: The Simulink model of the UAV flight dynamics. "yout.mat" contains the simulated sensor data and "xout.mat" contains the truth data of the flight.

ates modeled sensor data. The gyroscopes are modeled using

$$\text{gyro}_x = p + \sigma_{\text{gyro}}\xi,$$
$$\text{gyro}_y = q + \sigma_{\text{gyro}}\xi,$$
$$\text{gyro}_z = r + \sigma_{\text{gyro}}\xi,$$

where p, q, and r are the angular rates coming from the true UAV angular rates in "xout.mat", σ is the gyroscope noise variance parameter set in "param.m", and ξ is a zero mean normally distributed random number. The accelerometers are modeled by

$$\text{accel}_x = F_x + F_{\text{throttle}} + \sigma_{\text{accel}}\xi,$$
$$\text{accel}_y = F_y + \sigma_{\text{accel}}\xi,$$
$$\text{accel}_z = F_z + \sigma_{\text{accel}}\xi,$$

where F_x, F_y, and F_z are the aerodynamic forces acting upon the aircraft calculated using the properties of the aircraft defined in "param.m", and F_{throttle} is the force applied by the throttle. Finally, the GPS is modeled by

$$\begin{aligned} pn_{\text{GPS}} &= p_n + \sigma_{\text{GPS}}\xi, \\ pe_{\text{GPS}} &= p_e + \sigma_{\text{GPS}}\xi, \\ pd_{\text{GPS}} &= -p_d + \sigma_{\text{GPS}}\xi, \end{aligned}$$

$$\begin{aligned} n_{vel} &= (pn_{\text{GPS}} - pn_{\text{prev-GPS}})/Ts_{GPS}, \\ e_{vel} &= (pe_{\text{GPS}} - pe_{\text{prev-GPS}})/Ts_{GPS}, \\ d_{vel} &= (pd_{\text{GPS}} - pd_{\text{prev-GPS}})/Ts_{GPS}, \\ \psi_{\text{GPS}} &= \psi + \sigma_{\text{GPS}}\xi, \end{aligned}$$

where p_n, p_e, and p_d are the true position, and ψ is the true heading. The GPS velocities are calculated from noisy positions so no extra noise is added. They also use the GPS sample time to find the measured velocity. In each of the previous equations, noise characteristics similar to those of real measurements are used. These simulated measurements are chosen to most closely model the sensors that are used on the MERS aircraft.

Unfortunately, the MERS aircraft is considerably different from the aircraft modeled by this flight simulator. The UAV is considerably lighter, smaller, and slower than the MERS aircraft. This means that the angle of attack, aerodynamic coefficients, turning rate, rate of climb, inertial moments, lift, and drag of the UAV are very different from the MERS aircraft. Sideslip characteristics are very different because the UAV is much lighter and moves more slowly while not having a rudder. Because the MES aircraft is larger and moves more quickly with a rudder, it does not suffer from as much sideslip and has a lower angle of attack. These differences certainly affect the SR-UKF's performance, but the simulator is still useful to provide an assessment of the functionality of the SR-UKF.

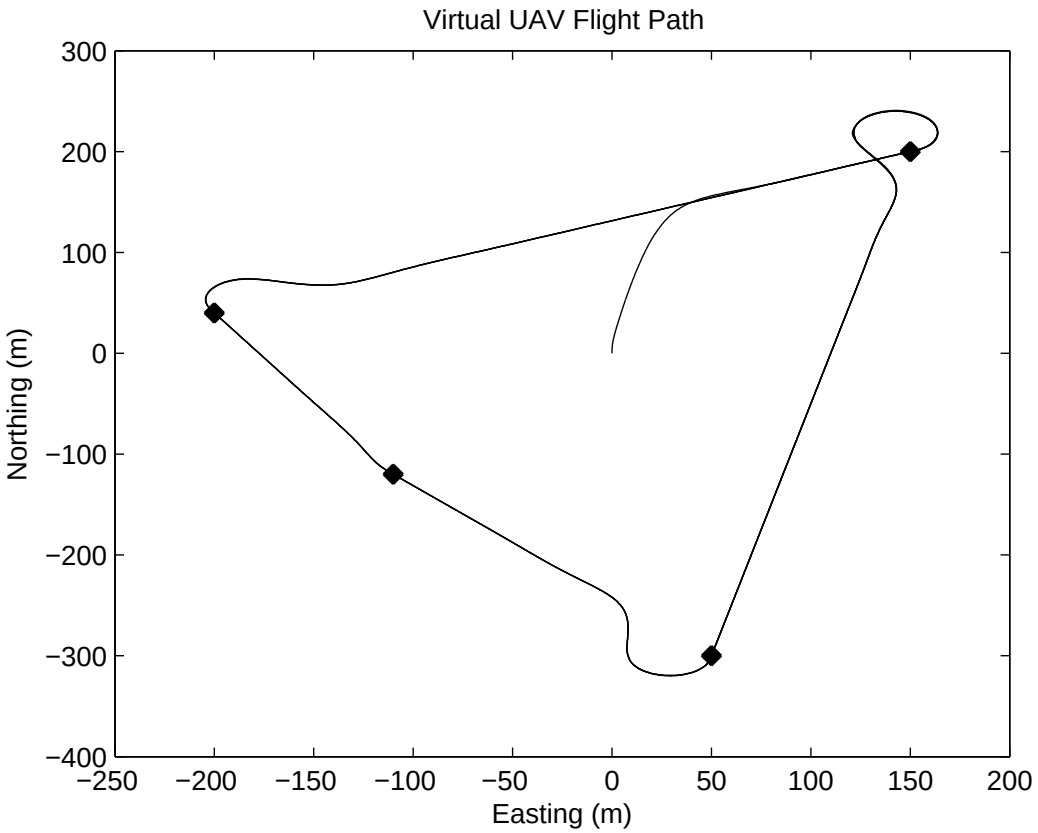

Figure 5.3: The flight path of the virtual UAV. The UAV starts at the center and flies a clockwise path around four different waypoints several times.

5.2 Kalman Filtering the Simulated Data

Once the simulator has been initiated, it directs the virtual aircraft to fly the path shown in Fig. 5.3. As the UAV flies, the computer-generated data and observation time are recorded.

After running the flight simulator for a few minutes (allowing the UAV to fly several rounds), and collecting a set of simulated sensor measurements, the Kalman filter is applied to the recorded data. The following set of graphs illustrate the truth data, the output of the Kalman filter, and the Kalman filter error for each of the states in the filter.

Figure 5.4 shows the position measurements of the SR-UKF. The figure is divided into three groups of three graphs, northing, easting, and altitude. The first of

the three graphs shows the truth data, the second shows the SR-UKF solution, and the third is the error, all of which are represented in meters. It is clear that the SR-UKF and truth data follow a very similar trend. The SR-UKF solution is generally less than 1 meter away from the actual position for each of the three position measurements. Table 5.1 gives some important statistics about the SR-UKF error. The mean error is close to zero, and the maximum error is close to 1 meter. By observing the altitude data, it is easy to see that the small UAV in this flight jumps and dips considerably (dropping 2 meters within 2 seconds) which significantly effects the accuracy of the SR-UKF. This is a rather extreme flight pattern that would not normally occur on the MERS aircraft; nonetheless, the SR-UKF does a reasonable job of tracking the position despite the somewhat erratic simulated aircraft platform behavior.

Table 5.1: Northing, Easting, and Altitude Errors (m)

Measurement	Mean Error	RMS Error	Standard Deviation	Max Error
Northing	0.0277	0.5073	0.5066	1.0130
Easting	-0.0042	0.3633	0.3633	1.1945
Altitude	-0.0035	0.0618	0.0617	1.2412

Figure 5.5 illustrates the body frame velocity output of the filter. As in Fig. 5.4, the figure is divided into three parts showing the u, v, and w SR-UKF measurements with respect to the truth. The body frame velocity is one of the most difficult states to track accurately due to the lack of any direct measurement of speed in the body frame. Neglecting sideslip, one of the assumptions previously made, is clearly not the best decision here. Ideally, v is always zero in the absence of sideslip, but we can see from Fig. 5.5 that v is clearly not always close to zero throughout the flight. This assumption negatively affects the output of the SR-UKF, however, with proper Q-tuning, these errors can be somewhat reconciled as is shown in the output of the SR-UKF. The other assumption, that u and w follow a constant angle of attack, is a better assumption, but still not ideal. Due to the UAV's behavior, the body frame velocity is under constant change, shown by the large spikes in u and w. While these

spikes are certainly difficult to track, they are not large enough to cause significant problems for the SR-UKF. The error statistics are given in Table 5.2. The mean error in w is caused by the virtual measurement of the angle of attack in the Kalman filter. While this virtual measurement may cause a small mean error in w, it is beneficial overall to the SR-UKF. As expected, the greatest maximum error is in v.

Table 5.2: Body Frame Velocity Errors (m/s)

Measurement	Mean Error	RMS Error	Standard Deviation	Max Error
u	0.0562	0.1243	0.1109	0.7704
v	0.0067	0.2980	0.2980	5.1275
w	-0.2591	0.4998	0.4274	2.7526

Figure 5.6 shows the non-centripetal acceleration measurements and truth data in the x, y, and z directions from top to bottom respectively. A large amount of error is expected in the acceleration measurements. Here we assume that the non-centripetal acceleration is zero, which of course is not true. Because of the large acceleration component in the simulated z axis, the measured acceleration (assumed to be close to zero) and the actual acceleration have large differences. However, this assumption helps to keep the velocity and position in check. If the acceleration is allowed to drift from zero too much, it has extreme effects on position and velocity. So, as shown in all three acceleration measurements, the non-centripetal acceleration is always close to zero. The actual acceleration has very large spikes which tend to cause large aberrations in other measurements such as attitude and body frame velocity (as shown earlier). These spikes are due to the extreme nature of the simulator output for the UAV's flight pattern. Table 5.3 illustrates the acceleration errors in each direction.

Figure 5.7 illustrates the measurements of the roll, pitch, and yaw angles and their errors. Firstly, it is easy to see that the SR-UKF estimates track all of the trends of the truth data. Roll is the most difficult to track accurately because it depends greatly upon the body frame velocity measurements and sideslip. If there is significant

Table 5.3: Non-centripetal Acceleration Errors (m/s^2)

Direction	Mean Error	RMS Error	Standard Deviation	Max Error
x	0.0659	0.3209	0.3141	2.8365
y	-0.0029	0.3112	0.3112	5.4124
z	0.0113	2.2606	2.2607	27.5543

sideslip (which is assumed to be zero in the SR-UKF model), it introduces problems for the SR-UKF. From Fig. 5.7, ϕ has the greatest error when the UAV takes a large turn. During a coordinated turn, when an aircraft banks to turn, it must roll to the right or left. It is also during a coordinated turn that sideslip is introduced; in the case of a small UAV without a rudder, the sideslip can be severe. Sideslip coupled with extreme acceleration spikes can be disastrous to the flight model, however, the SR-UKF still performs very well in tracking these measurements. Statistical properties of the attitude errors are given in Table 5.4. It is also important to note that the mean error in θ is due to the constant angle of attack assumption made in the synthetic measurements within the SR-UKF. Pitch is closely related to w, so due to the non-zero mean error in w, a non-zero mean error is expected in θ. While the angle of attack assumption causes this discrepancy, it is still valuable to the SR-UKF as a whole. Finally, the heading is kept in check by the GPS heading measurement. The assumption that GPS heading and yaw angle are the same is only true in a zero wind environment. The simulator does not introduce any wind into the UAV's flight environment, so this assumption is better in simulation than it is in an actual flight.

Table 5.4: Attitude Errors(°)

Angle	Mean Error	RMS Error	Standard Deviation	Max Error
ϕ (Roll)	-0.0686	3.4242	3.4237	21.5885
θ (Pitch)	1.3137	2.5649	2.2031	19.1165
ψ (Yaw)	0.0170	0.8090	0.8088	6.4115

Figure 5.8 shows the measurements and truth data related to the angular rates p, q, and r. It is clear from the graphs that p, q, and r are tracked quite accurately. This is due to the fact that they are directly measured by the onboard gyroscopes. On the MERS aircraft, the gyroscopes are very accurate, and so the simulated gyroscopes are set to be very accurate as well. The most significant error in angular rate is found in the measurement of q. This is because of the extreme acceleration and pitching of the aircraft discussed previously. The error however, is still relatively small. Statistical analysis of the errors are given in Table 5.5.

Table 5.5: Angular Rate Errors (rad/s)

Measurement	Mean Error	RMS Error	Standard Deviation	Max Error
p	-5.613×10^{-5}	3.0711×10^{-4}	3.0196×10^{-4}	0.0084
q	-0.0018	0.0044	0.0040	0.0462
r	-3.1391×10^{-4}	8.8519×10^{-4}	8.2770×10^{-4}	0.0065

Overall, the SR-UKF tracks the truth data relatively accurately. Despite the fact that the simulated measurements frequently have large spikes, the SR-UKF is able to track the states through their nonlinear measurement functions. Many of the approximations discussed previously, such as the angle of attack, sideslip, and non-centripetal acceleration assumptions, are not as valid in the case of a small aircraft. Sideslip significantly affects the accuracy of the results, and because smaller aircraft without a rudder suffer from significantly more sideslip than larger aircraft do, we see greater errors in the output of the Kalman filter than would occur otherwise. Despite the somewhat extreme nature of the simulated flight, the SR-UKF is able to give reasonably low-error estimates of attitude and position.

5.3 Kalman Filtering Actual Flight Data

With the SR-UKF shown to work on the simulated data, it is applied to real data from the MERS aircraft. After motion data from a SAR-recording flight is collected with the MOTRON, the sensor recordings are prepared using the scaling

factors and GPS to northing/easting conversion. The SR-UKF is then applied to the data. The following sections discuss the overall flight measurements and a smaller section of the flight that seemed to most accurately meet the assumptions made by the SR-UKF.

5.3.1 Measurements of the Entire Flight

The following data set is taken from the time the aircraft left the ground to the time it touched down, or in other words, only during flight. This set of data is used because as the plane taxis on the runway, it violates the assumption of a coordinated turn and has no angle of attack. Fig. 5.9 shows the overhead view of the MERS aircraft flight. From the figure, we can see that the plane flies several circles over the Brigham City area and then returns to the Brigham City Airport runway.

Figure 5.10 shows the position measurements of the MERS aircraft from takeoff to touchdown. The top three figures show northing measurements provided by the SR-UKF, the NovAtel system, and their difference. The second set gives the easting measurements and the third gives altitude. The plane starts at an altitude of 1275 meters (the altitude of the Brigham City airport), reaches an altitude of 1659 meters, takes a dip to 1460 meters, then returns to 1650 meters before returning and landing after a 20 minute flight. The position measurements of the SR-UKF are within 50 cm of the NovAtel solution. Table 5.6 provides statistical data for the difference between the NovAtel solution and the SR-UKF solution.

Table 5.6: Difference Between SR-UKF and NovAtel Position (m)

Measurement	Mean	RMS	Standard Deviation	Max
Northing	-0.0172	0.1997	0.1989	0.4646
Easting	0.0129	0.1502	0.1497	0.4780
Altitude	0.0028	0.0466	0.0465	0.4584

Figure 5.11 shows the NovAtel and SR-UKF solutions for roll, pitch, and yaw respectively. From the figures it is clear that the SR-UKF and NovAtel systems share

common trends. Table 5.7 gives data about the error between the NovAtel and SR-UKF systems. Several assumptions about the aircraft's flight account for some of the difference. The assumption of constant angle of attack does not allow the SR-UKF solution to grow in pitch very much because the virtual measurement of the body frame velocity (particularly u and w) pull it down. This assumption does not allow the SR-UKF to achieve the higher angles of pitch that may have existed in the actual flight, and that the NovAtel system may have reported. These discrepancies can be found between times 0-200 and 400-600. Furthermore, the assumption of zero sideslip contributes to some of the error in the roll angle. Because sideslip is not likely zero in reality, whenever there exists non-zero velocity in the v direction, there is an error in the roll angle. This is evident at times 100, 300, 1000, and 1300. Also, because there is no direct measurement of the body frame velocity or attitude, and because the NovAtel implements a smoother, the SR-UKF solution is significantly noisier than the NovAtel solution. Because of these factors, the greatest difference is found in the roll angle.

Table 5.7: Difference Between SR-UKF and NovAtel Attitude(°)

Measurement	Mean	RMS	Standard Deviation	Max
ϕ(Roll)	0.1250	3.1797	3.1773	16.4655
θ(Pitch)	0.1841	0.9416	0.9235	3.6347
ψ(Yaw)	0.3515	0.4980	0.3527	1.5411

Figure 5.12 shows the remaining states found in the SR-UKF. The body frame velocity measurements give a good metric as to the validity of our angle of attack and sideslip assumptions. From the body frame velocity (u, v, and w), we can see that most of the overall velocity is in the u direction as we would expect. It is also clear that the assumption of zero sideslip was not completely accurate from the v measurement, as it clearly drifts away from zero. Finally, w has an almost constant velocity over the flight, which gives credence to the constant angle of attack assumption. The non-centripetal acceleration is nearly zero as we would expect, and the p, q, and r

measurements seem reasonable. There is no NovAtel solution for these states, so no error metrics are given.

Overall, the SR-UKF performs much better on the actual flight data than it did on the simulated flight data. This is due to the fact the assumptions discussed earlier are much more valid for larger aircraft than the small UAV simulated case. The results are very encouraging.

5.3.2 A Smaller Portion of the Flight

After viewing the measurements taken from the entire flight, it is valuable to observe a set of data taken from a flight that more accurately reflects the assumptions that the SR-UKF makes (no sideslip, constant angle of attack, no wind, etc.). This subsection discusses a smaller 10 minute portion of the MERS aircraft flight that closely meets these conditions.

Figure 5.13 illustrates the northing, easting, and altitude measurements of the SR-UKF compared to the NovAtel system. As the figure shows, the position tracking is as accurate as the NovAtel system to within 40 cm. This is a better result than in the simulation or on the full set of flight data. Also, there is a significant improvement in the altitude measurement. Because this portion of the flight had very good conditions, the position tracking is excellent. Table 5.8 shows some statistical differences between the SR-UKF solution and NovAtel solution.

Table 5.8: Difference Between SR-UKF and NovAtel Position (m)

Measurement	Mean	RMS	Standard Deviation	Max
Northing	0.0080	0.2063	0.2061	0.4350
Easting	-0.0316	0.1528	0.1495	0.3939
Altitude	0.0032	0.0435	0.0434	0.2534

Figure 5.14 shows the result of the SR-UKF versus the NovAtel for this 10 minute flight section. The roll difference looks much more white than before and never wanders above 8 degrees. Considering that the NovAtel data is smoothed, the SR-

UKF performs very well. All three of the Euler angle measurements have a very small difference with respect to the NovAtel solution. Table 5.9 contains the statistical data about this difference. It is important to note that the SR-UKF does not allow the pitch angle to stray very far above 5 degrees. This is due to the assumption that the angle of attack is constant. At 50 seconds into this flight section we can see that the NovAtel pitch climbs higher than the SR-UKF is allowed to. The NovAtel solution is probably more accurate because it does not rely on the angle of attack assumption that the SR-UKF does. However, the SR-UKF is still only a degree or two away from the NovAtel solution. This section of data from the SR-UKF is very encouraging, and the data shows that under the right conditions, the SR-UKF performs very well.

Table 5.9: Difference Between SR-UKF and NovAtel Attitude(°)

Measurement	Mean	RMS	Standard Deviation	Max
ϕ(Roll)	0.2268	2.2230	2.2115	10.0244
θ(Pitch)	0.2586	0.8876	0.8492	2.5067
ψ(Yaw)	0.3484	0.5591	0.4372	1.5505

Figure 5.15 shows the remaining states of the 10 minute section. These measurements have nothing to compare with, but are given to show that they are reasonable. The SR-UKF performed much better on this set of actual flight data. This is due to the fact that the aircraft more closely met the approximations discussed previously.

5.4 Filtered Results

It is clear that even after Kalman filtering the optimal data, the output of the SR-UKF is still too noisy, and as Fig. 5.16 shows, there is significant harmonic distortion in the roll angle estimate. These harmonics are also found in the raw gyroscope and accelerometer data and are probably due to the intrinsic harmonic response of the aircraft.

In order to correct these distortions, a low pass filter is used on ϕ and θ. A high order filter is found to be the most effective and is shown in Fig. 5.17. An equiripple FIR filter of order 1126 is chosen using the Parks-McClellan method to lower the noise and all the harmonic content. A very long filter is needed in order to isolate the low frequency content of interest, 0.001 to 0.005 πrad/sample while keeping the side lobes down to a low level. Although this filter is long, it is less than 1/100 of the size of the roll estimate. However, a filter of this length would be unreasonable to use in real time applications.

Fig. 5.18 shows the resulting attitude estimates after applying the low pass filter. As before, the NovAtel solution is given first, followed by the filtered SR-UKF solution and their difference for roll, and pitch respectively. No filtering is applied to the yaw angle due to the irregular 0-360° wrapping. Table 5.10 gives the new error statistics after filtering. As can be seen in the figure and table, the error is greatly reduced by the low pass filter. The maximum error has been reduced by half, the mean error is much closer to zero, and the graph looks much more similar to the NovAtel solution.

Table 5.10: Difference Between Filtered SR-UKF and NovAtel Attitude(°)

Measurement	Mean	RMS	Standard Deviation	Max
ϕ(Roll)	-0.0946	1.1863	1.1825	5.0120
θ(Pitch)	0.4372	0.8622	0.7431	1.7848
ψ(Yaw)	0.3484	0.5591	0.4372	1.5505

5.5 Conclusion

Overall, the SR-UKF does a reasonable job estimating the simulated UAV movement and performs very well on actual flight data. The SR-UKF created estimates within a few centimeters of the position and within a few degrees of the attitude measurements of the NovAtel solution. The SR-UKF performs similarly on data sets from additional flights made by the MERS aircraft. Most major discrepancies can

be accounted for in the approximations made to simplify the implementation of the SR-UKF. I found that the output of the SR-UKF is very sensitive to the covariance tuning, and even after the SR-UKF has run, additional low pass filtering was necessary. Most of the time I spent developing this SR-UKF was spent tuning the covariance matrix: incrementally making small changes to Q and testing the result. It is also important to note that if either the sensors or aircraft are changed, the parameters will need to be adjusted to account for the changes.

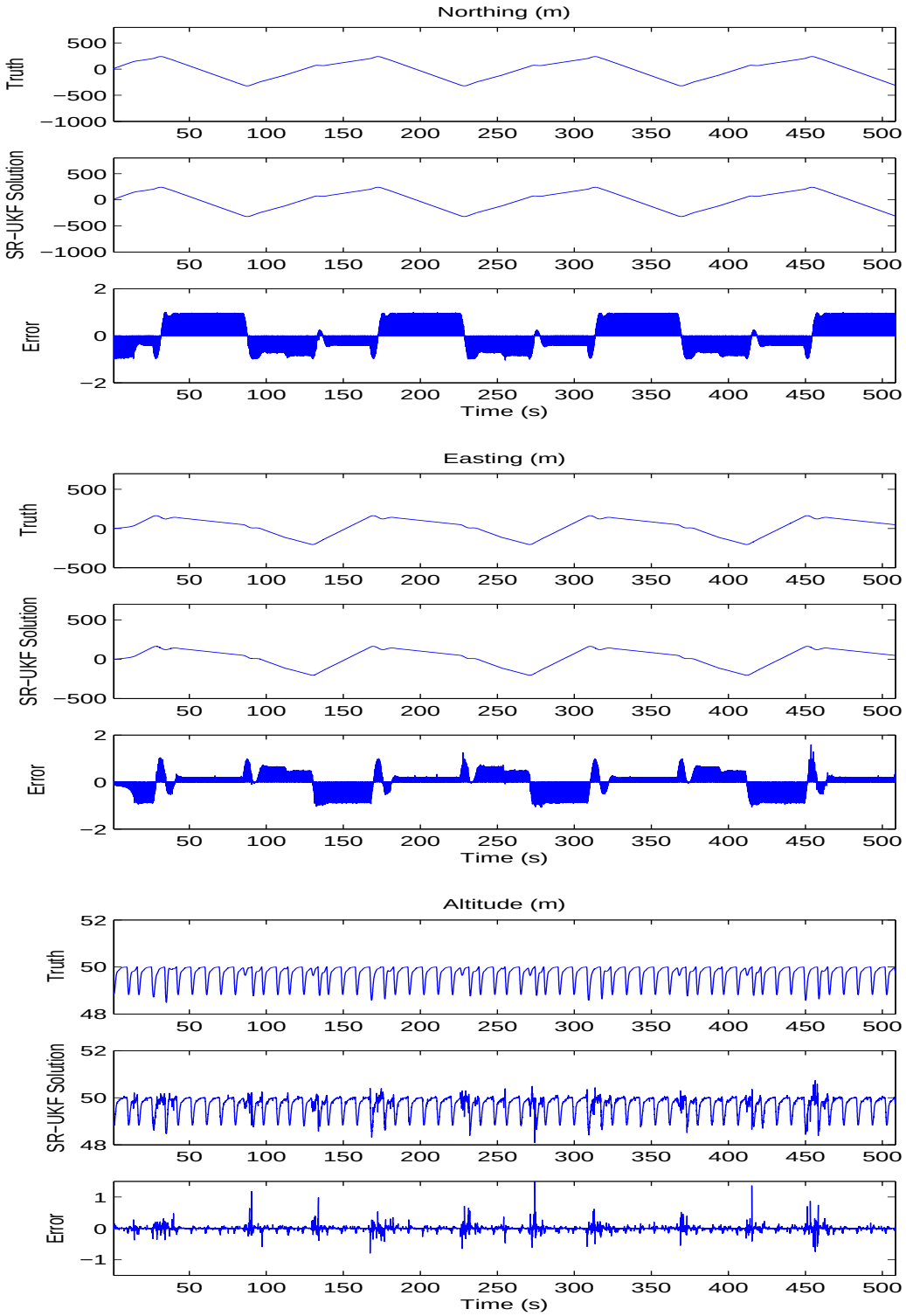

Figure 5.4: Truth data, SR-UKF solutions, and SR-UKF error for northing, easting, and altitude. See text and Table 5.1 for analysis.

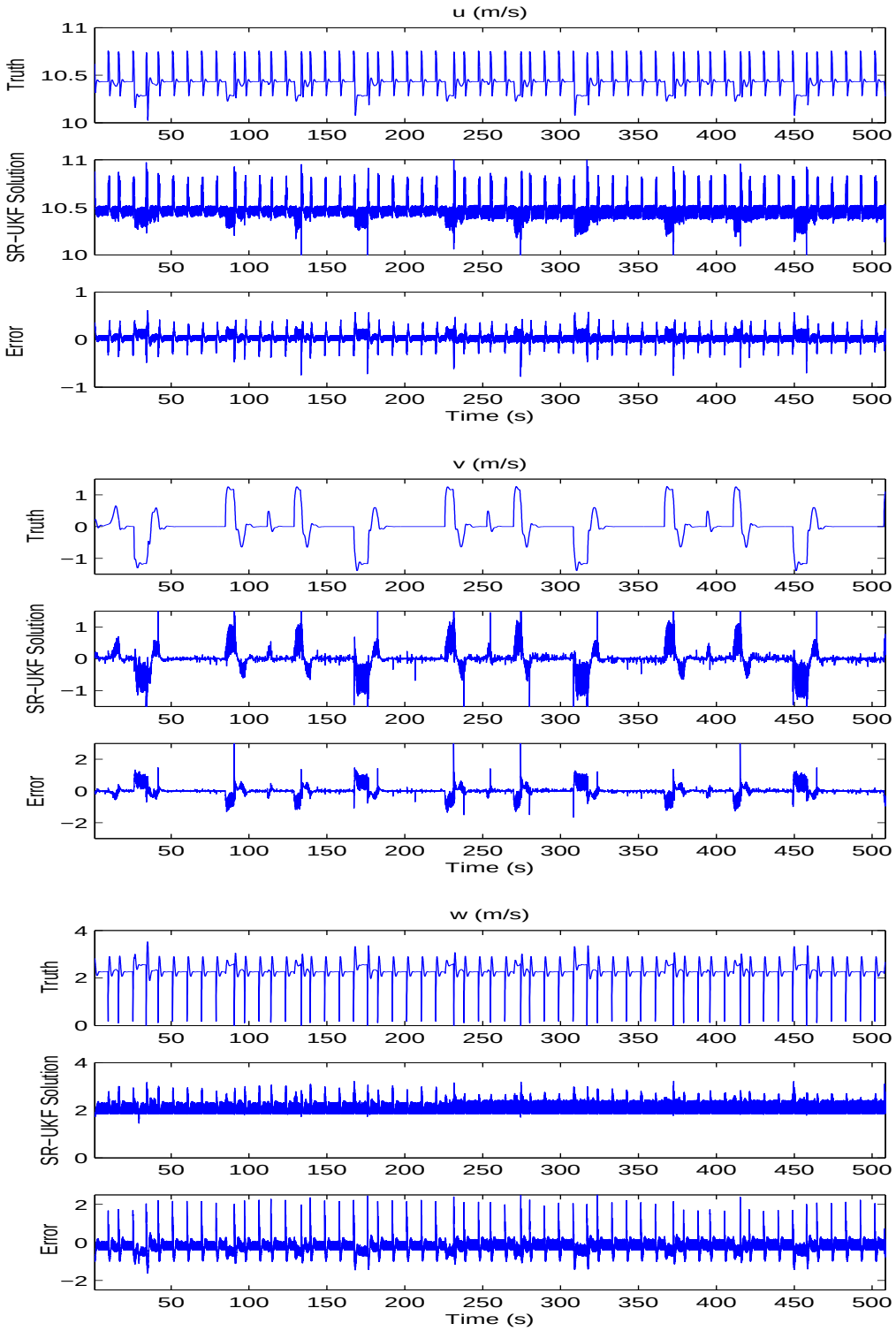

Figure 5.5: Truth data and the SR-UKF solutions for u, v, and w. See text and Table 5.2 for analysis.

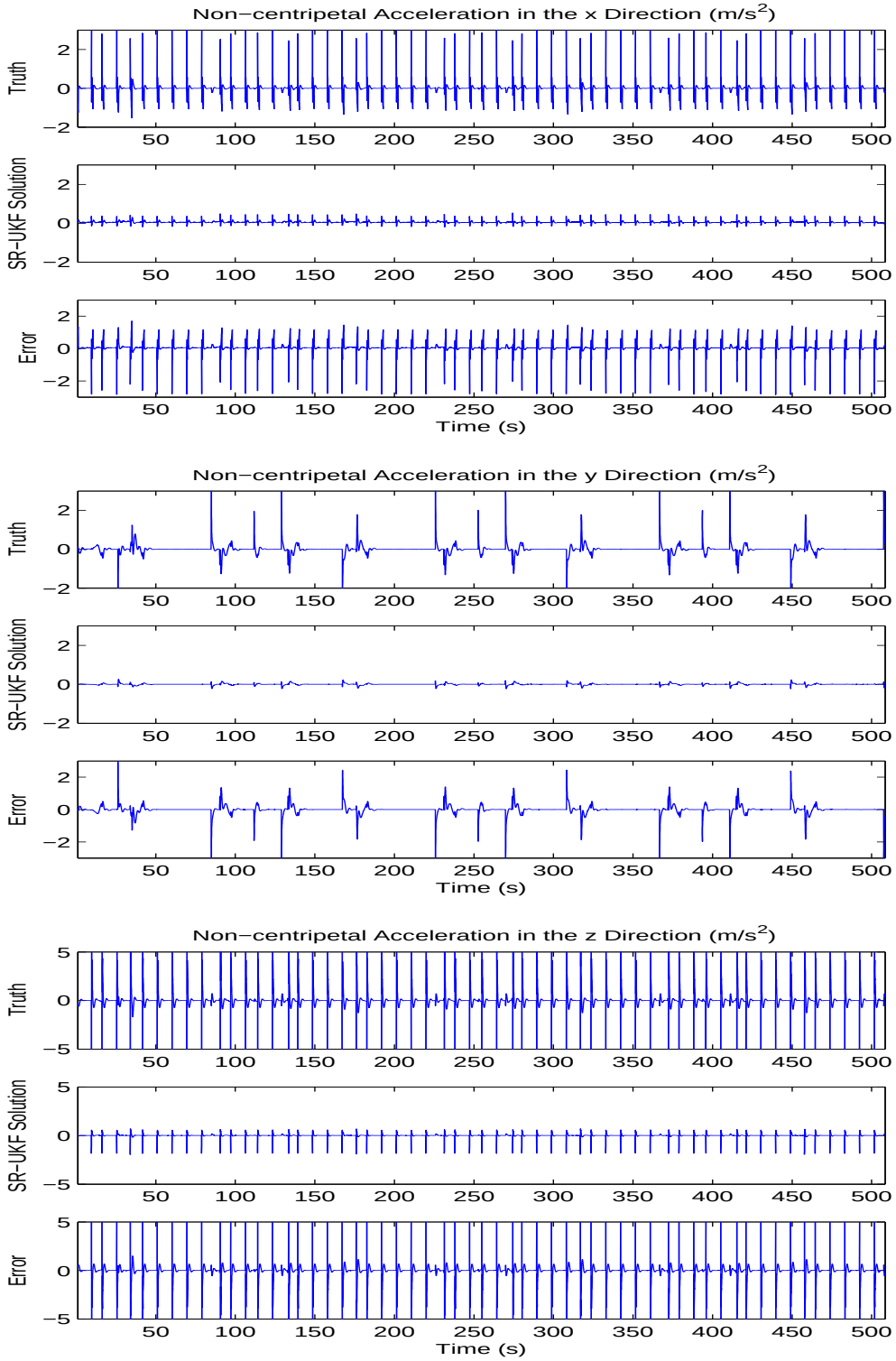

Figure 5.6: Truth data and the SR-UKF solutions for non-centripetal acceleration in the x, y, and z directions. See text and Table 5.3 for analysis.

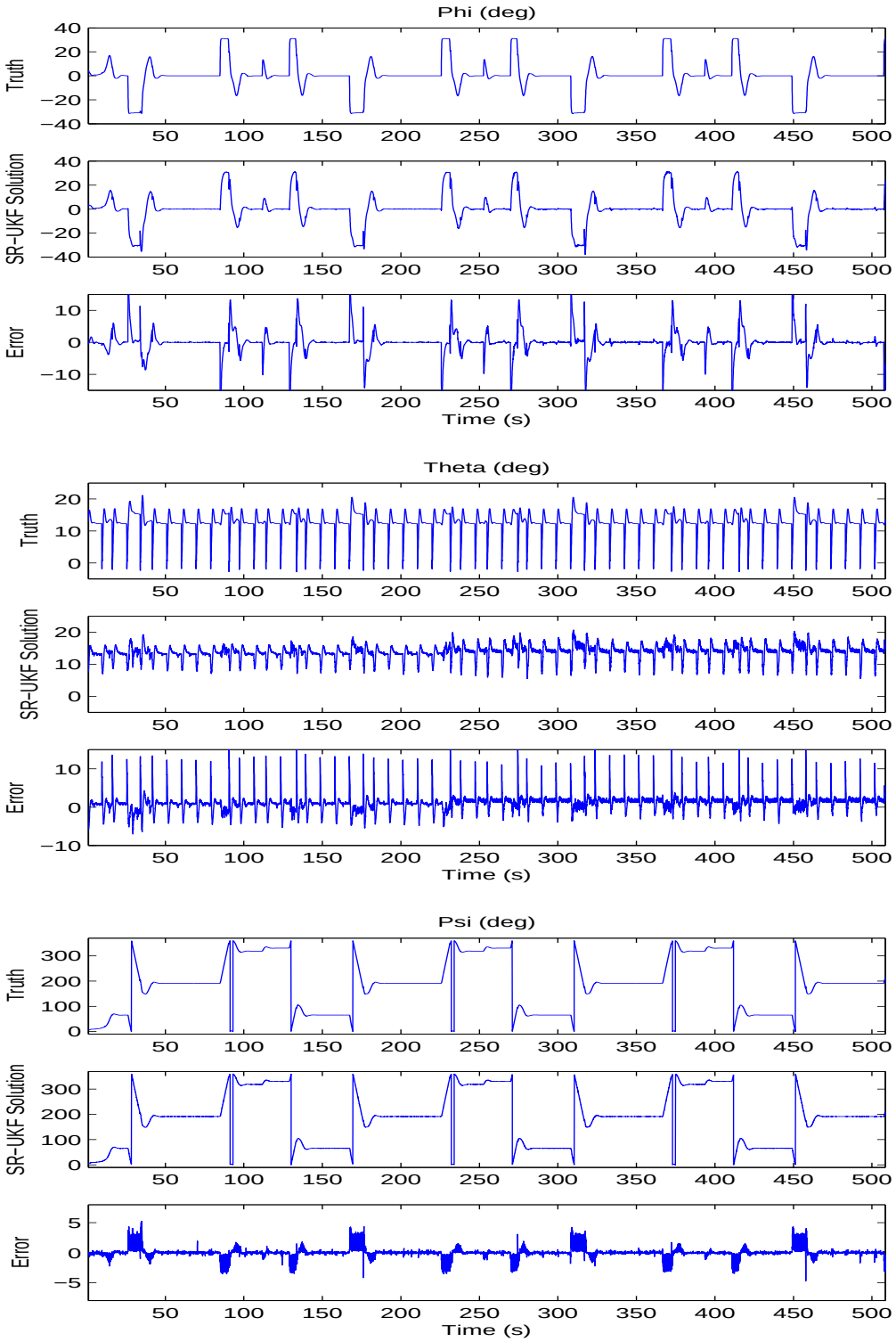

Figure 5.7: Truth data and the SR-UKF solutions for roll, pitch, and yaw. See text and Table 5.4 for analysis.

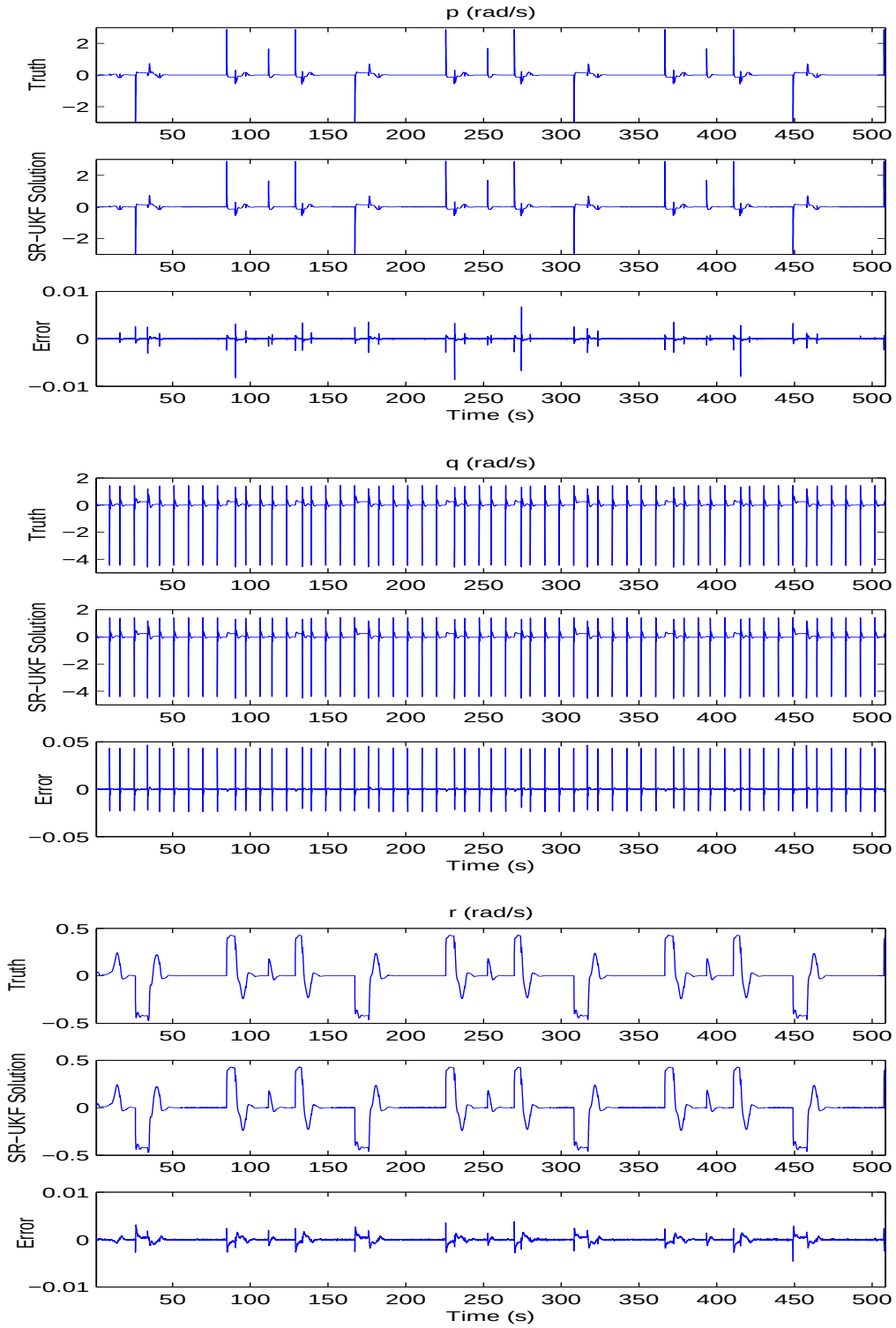

Figure 5.8: Truth data and the SR-UKF solutions for angular rates p, q, and r. See text and Table 5.5 for analysis.

Figure 5.9: The flight path of the MERS aircraft The drop pin indicates the starting position just before takeoff.

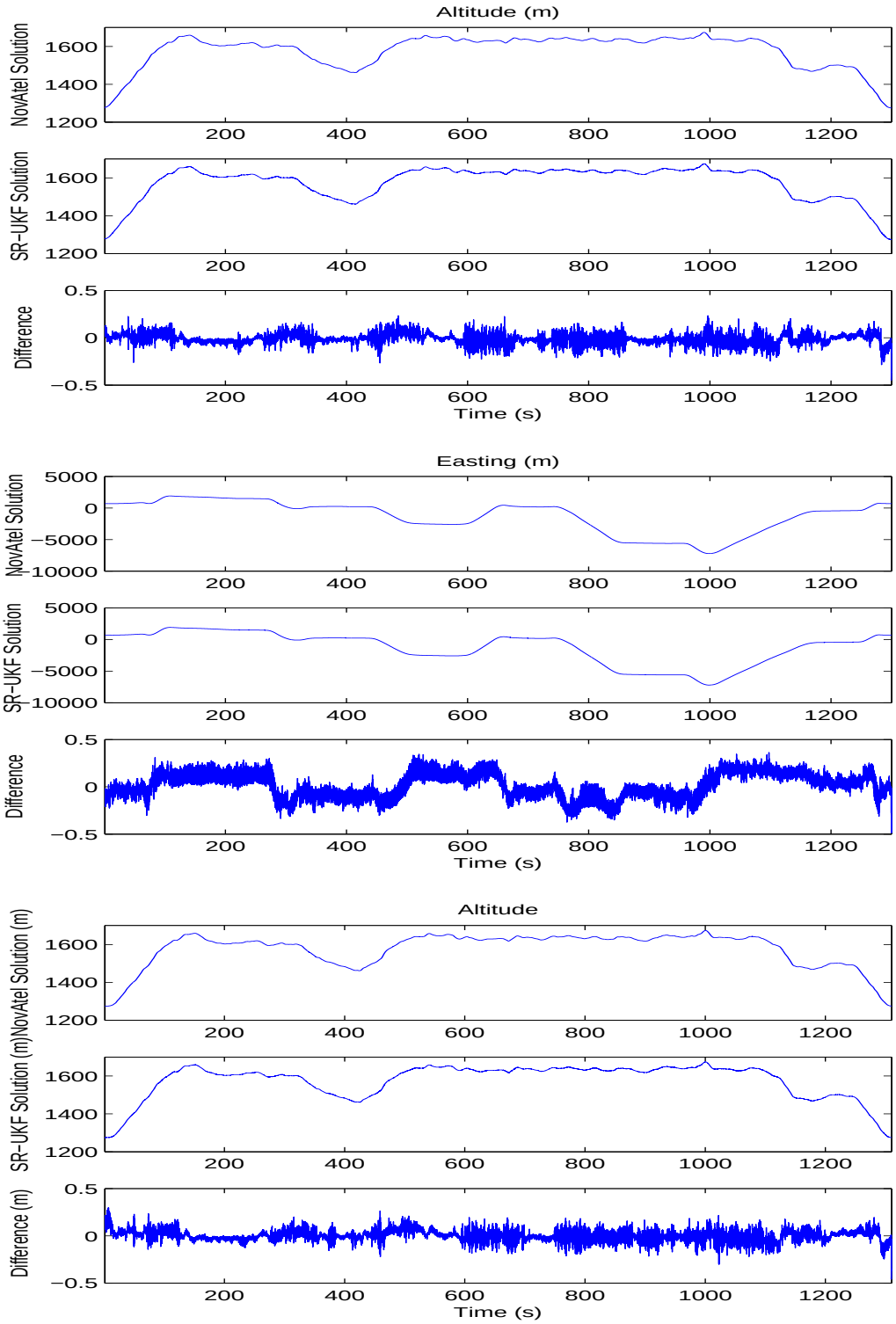

Figure 5.10: NovAtel solution and the SR-UKF solution for northing, easting, and altitude. See text and Table 5.6 for analysis.

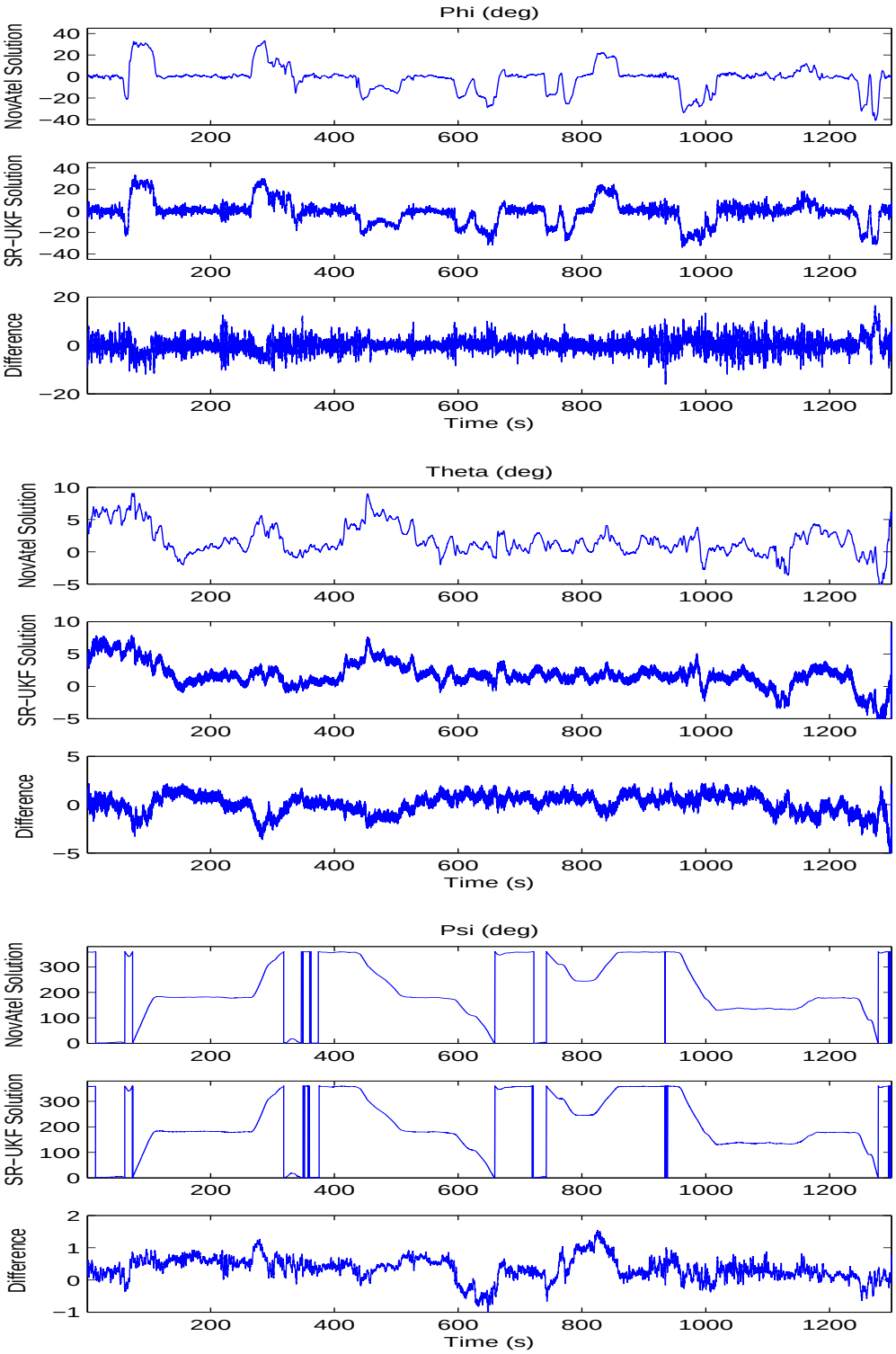

Figure 5.11: NovAtel solution and the SR-UKF solution for roll, pitch, and yaw. See text and Table 5.7 for analysis.

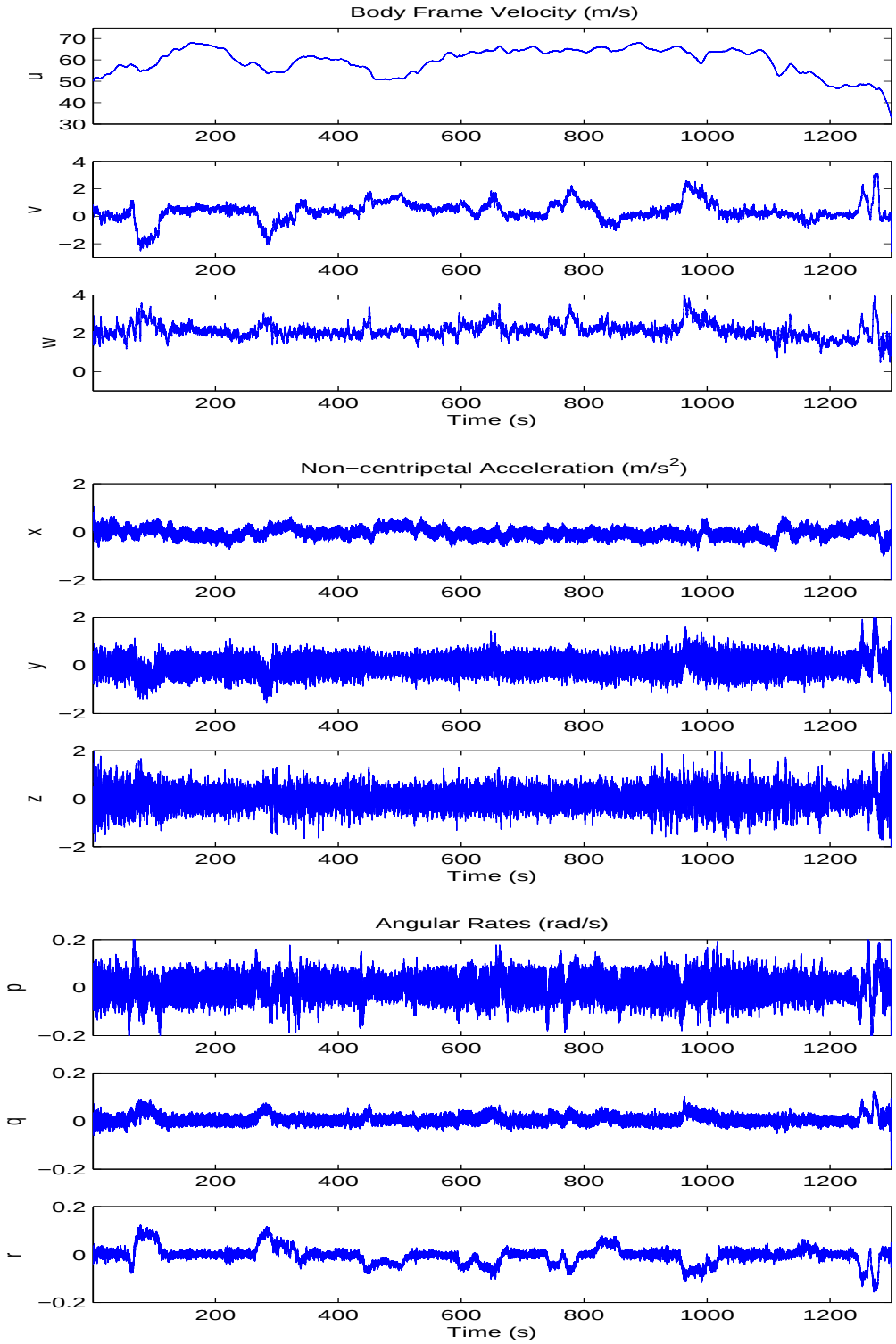

Figure 5.12: NovAtel solution and the SR-UKF solution for the body frame velocity, non-centripetal acceleration, and angular rates. See text for analysis.

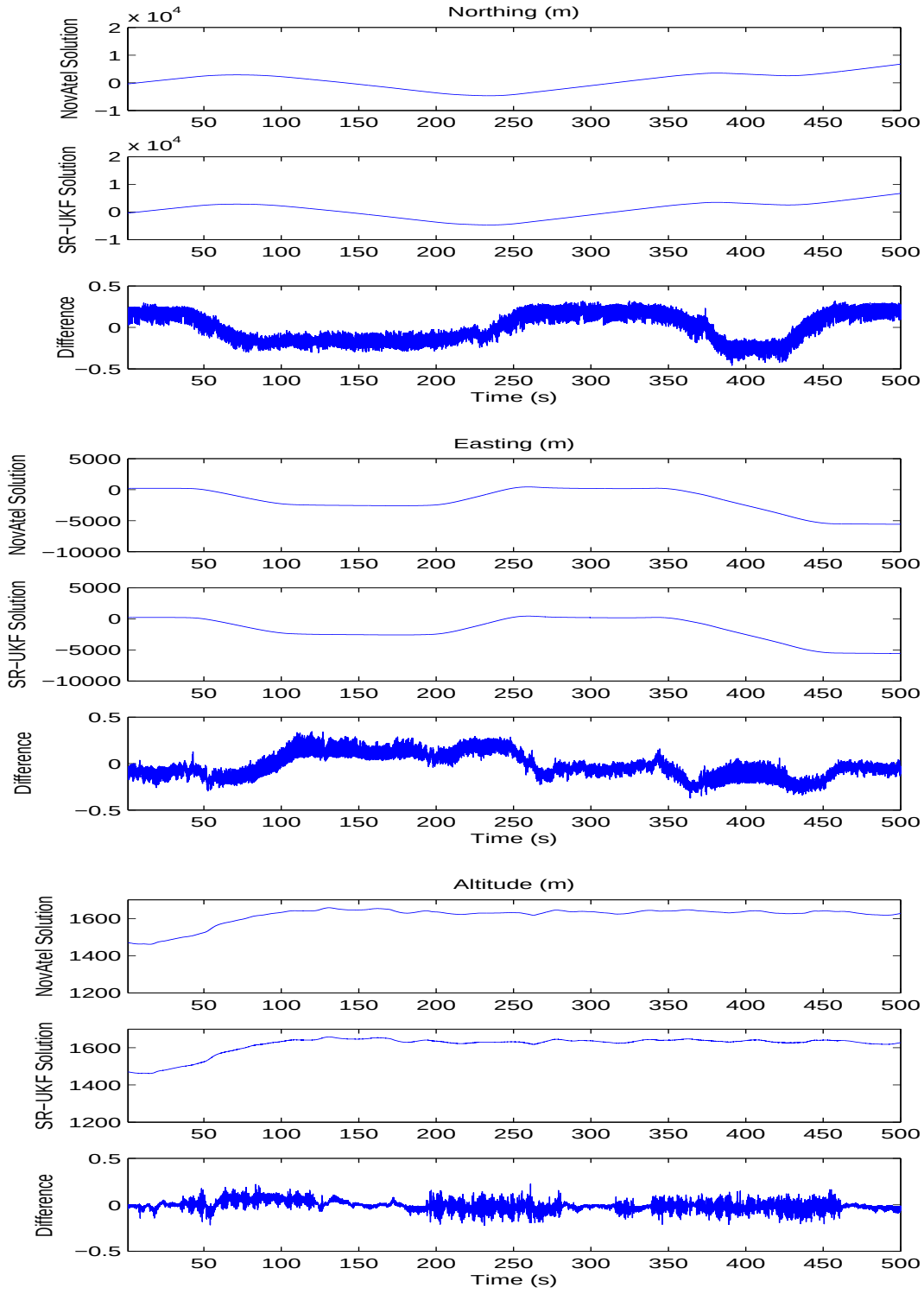

Figure 5.13: NovAtel solution and the SR-UKF solution for northing, easting, and altitude.

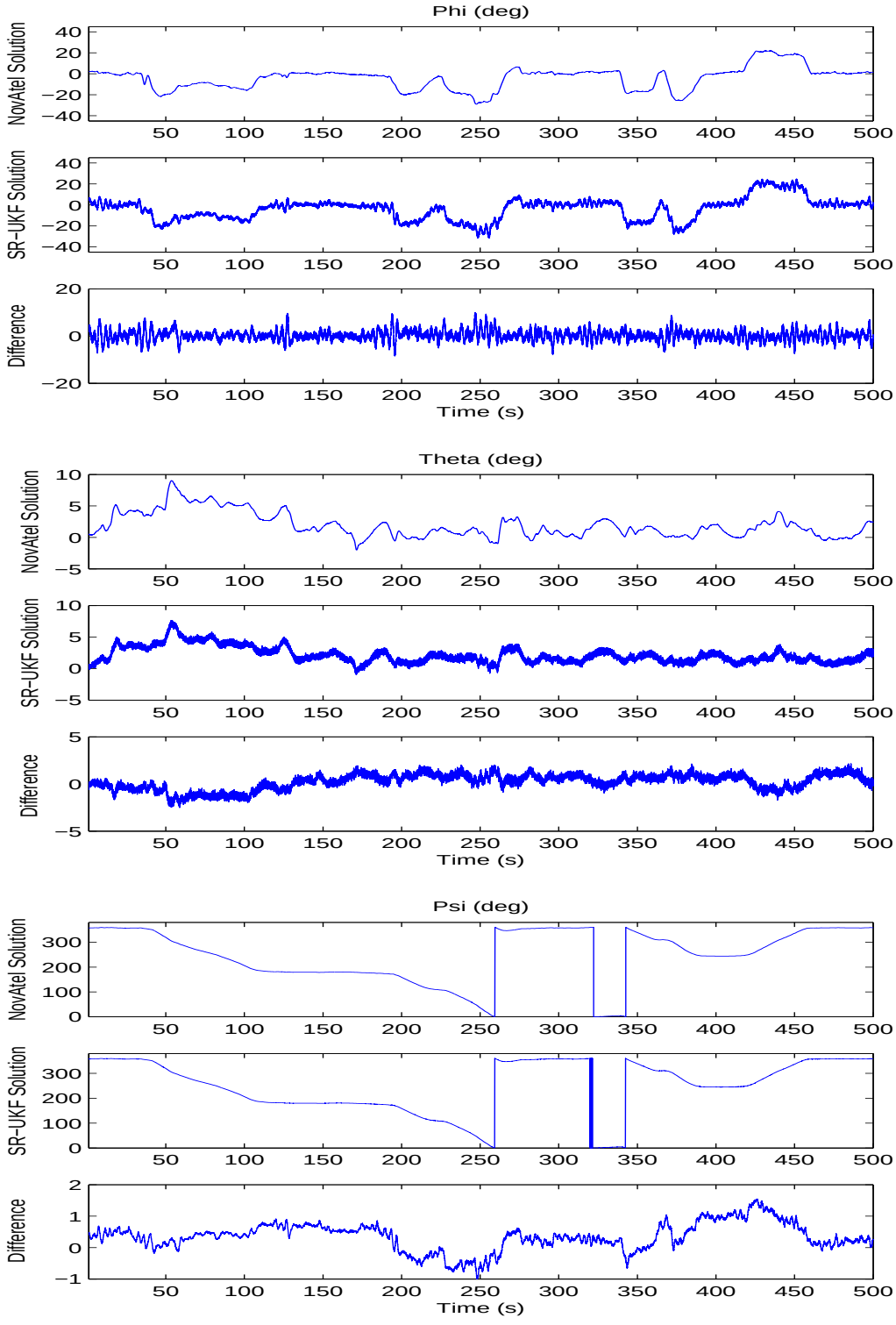

Figure 5.14: NovAtel solution and the SR-UKF solution for roll, pitch, and yaw.

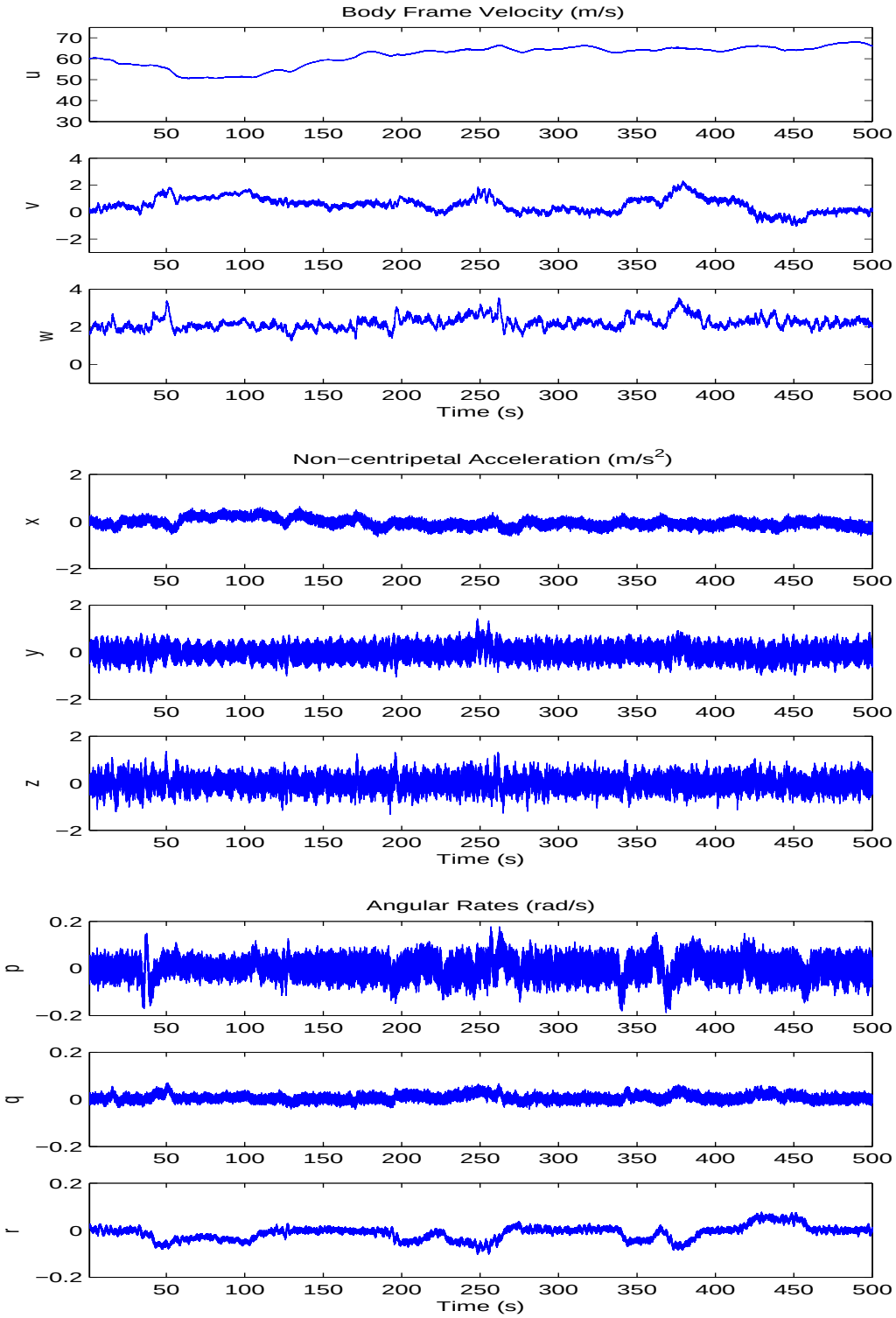

Figure 5.15: NovAtel solution and the SR-UKF solution for the body frame velocity, non-centripetal acceleration, and angular rates.

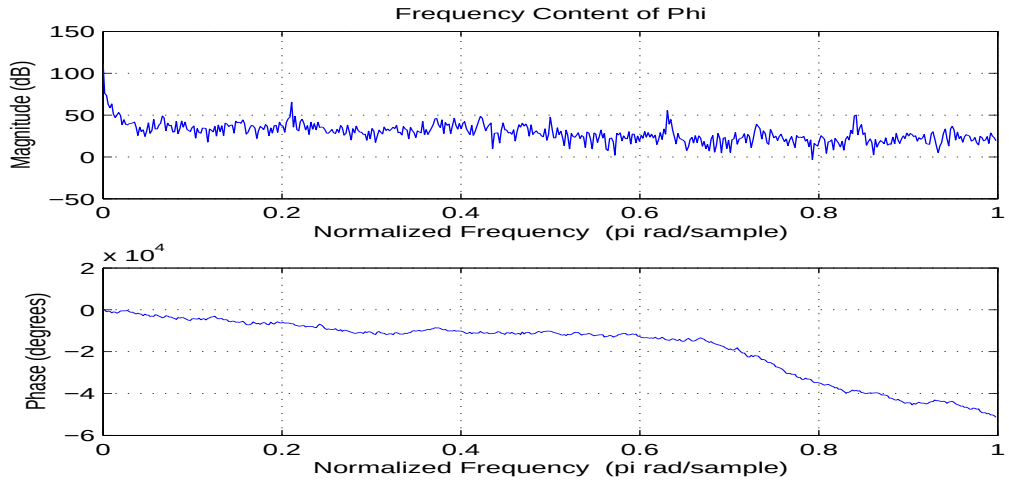

Figure 5.16: Frequency content of ϕ. Notice the strong harmonic content at 0.21π, 0.42π, 0.63π, and 0.84π rad/sample.

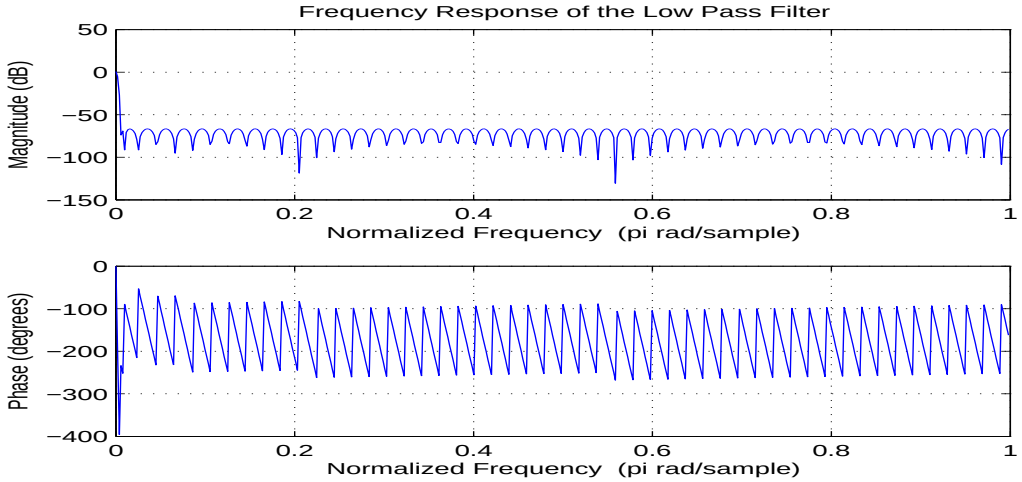

Figure 5.17: Frequency response of the low pass filter.

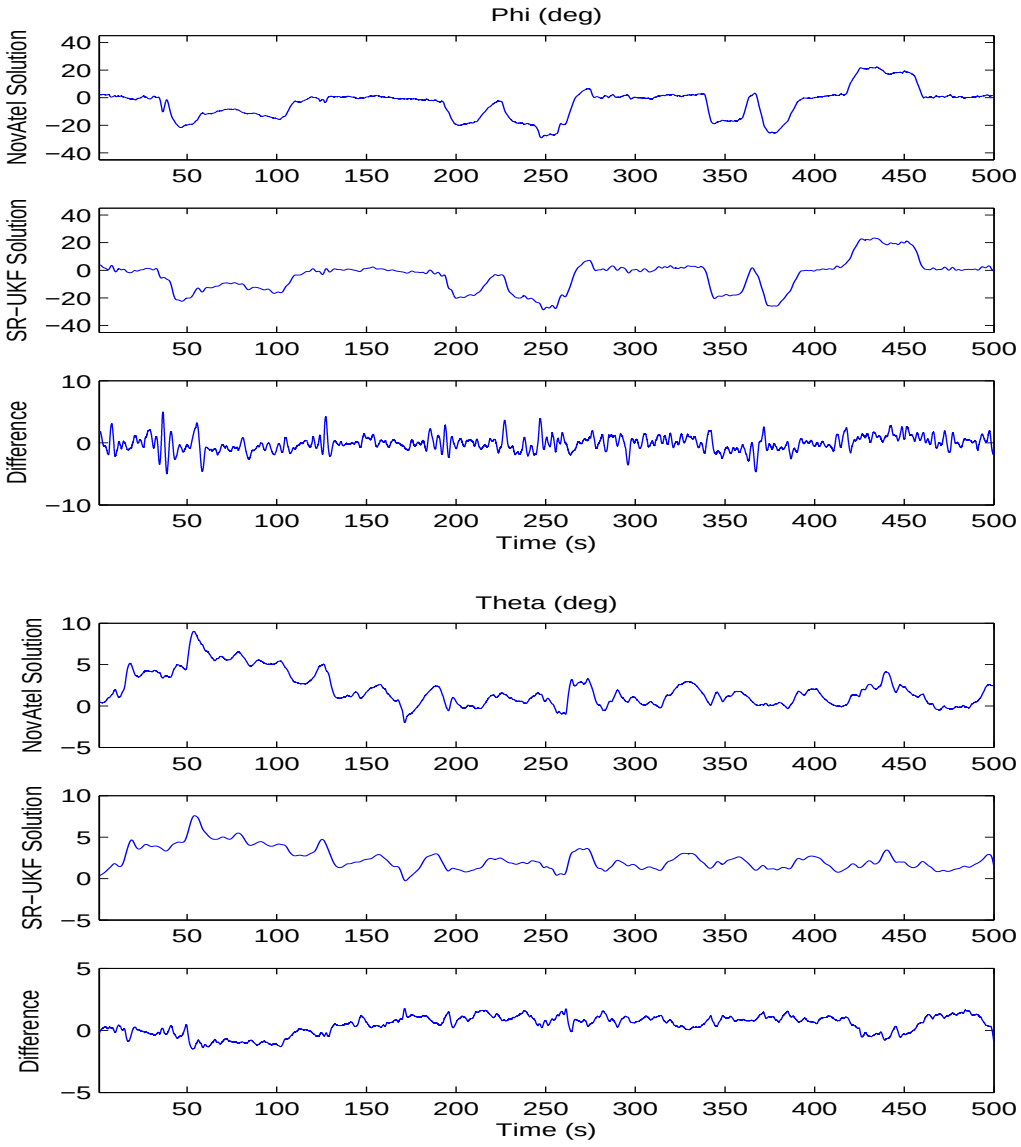

Figure 5.18: NovAtel solution and the low pass filtered SR-UKF solution for roll and pitch.

Chapter 6

Conclusion

The research provided in this thesis describes a method for estimating flight dynamics from measurements provided by laser gyroscopes, accelerometers, and high precision GPS integrated with a SR-UKF. The results of this research are thoroughly tested, examined, and compared to that of a high cost navigation system and are found to be surprisingly similar. Although the results of the SR-UKF are less accurate than the NovAtel system, they approach the accuracy of the NovAtel system. This result is surprising in that the development effort involved and cost of the SR-UKF is extremely low compared to the NovAtel system.

This thesis has also presented a straightforward model for fixed-wing aircraft flight dynamics, and has presented a simple, low-cost data recording scheme for high precision motion sensors. A SR-UKF was implemented using MATLAB software, and was optimized for rapid calculation. While MATLAB may run much more slowly than C/C++, the MATLAB code gives clear instruction on how to optimally program an SR-UKF for future use.

Due to the nature of SAR image processing, real time calculations are not always necessary. However, the dynamics estimation system here can be easily implemented in real time without the low pass filter. This is another benefit of the SR-UKF compared to the EKF. Because the SR-UKF runs more computationally efficient than the EKF, it is easier to execute in real time while at the same time gaining accuracy.

The main drawback of the SR-UKF described by this thesis is the flight dynamics assumptions. Assuming that sideslip is zero greatly simplifies the SR-UKF, but also contributes to inaccuracy in the body frame velocity and attitude. Furthermore,

assuming a constant angle to attack makes implementation easier, but the results less correct. With proper covariance tuning and a lot of patience, these assumptions could be done away with.

Overall, the research presented by this thesis has shown that a simple SR-UKF implementation can approach the effectiveness of an expensive and complex motion measurement system implemented with an EKF. In ideal conditions, the SR-UKF was nearly as good, and cost much less.

6.1 Contributions

The research presented in this thesis contributes the following:

- A user friendly standalone motion sensor measurement recording system that can be used with multiple sensors of any type (MOTRON).
- A state-space flight dynamics estimation model for fixed wing aircraft.
- An estimation model of three IMAR-IMU laser gyroscopes and accelerometers.
- A unique formulation of the square-root Unscented Kalman filter in which the mean value is not a traditional weighted sum, but simply the previous state mean passed through the nonlinear function f.
- A method for estimating initial attitude conditions of a stationary aircraft.
- A method of integrating simulated measurements into actual flight data to improve filter accuracy.
- A functional flight simulator that correctly models the NovAtel sensors onboard a small UAV.
- A fully functional flight dynamics estimation system.

6.2 Future Work

Areas in which future work may be done include:

- Elimination of flight dynamics approximations. By successfully eliminating the angle of attack, sideslip, coordinated turn, and acceleration assumptions, the estimation scheme can be applied to any free moving body. In other words, the estimation system can be used in automobiles, watercraft, helicopters, as well as fixed-wing aircraft. It would also make the Q-tuning independent of the platform, which would be very useful for not only SAR, but robotics and guidance systems.

- Use of quaternions. Using quaternions would eliminate the need for Euler angles, and hence, eliminate the problems associated with their use. Quaternions remove the chance of gimbal lock which occurs when two of the three axes of rotation in a three dimensional space are driven to the same direction resulting in the loss of one degree of freedom. By using quaternions, the motion estimation system is able to rotate freely without encountering this issue. It would also eliminate the issues that can occur with 0°-360° crossings.

- Implementation of a Kalman smoother. The SR-UKF estimates the state covariance at each time and measurement update. If each of those covariance updates were stored in memory, a Kalman smoother could be easily implemented. This would help to reduce the noise found in the results of the SR-UKF, and would be a much better alternative to a simple low pass filter.

- Verification of the SR-UKF using SAR imagery. If the data collected by BYU's nu-SAR were motion compensated correctly using the output of the SR-UKF, it would be a strong indication that the motion estimates were correct.

- Move the SR-UKF to operate in real time on the MOTRON. By optimizing the SR-UKF to run in C or C++, it could be greatly sped up and run in real time onboard the MOTRON, for real time dynamics estimates.

Bibliography

[1] R. van der Merwe and E. A. Wan, "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation," *Acoustics, Speech, and Signal Processing(ICASSP) IEEE Proceedings*, vol. 6, pp. 3461–3464, May 2001. xi, 1, 17, 18

[2] *SPAN for OEMV User Manual*, 4th ed., NovAtel Inc., 1120 - 68 Avenue NE, Calgary, Alberta, Canada T2E 8S5, 2007. 2

[3] F. T. Ulaby, R. K. Moore, and A. K. Fung, *Microwave Remote Sensing.* Norwood, MA: Artech House, 1981. 7

[4] R. W. Beard and T. W. McLain, "Guidance and Control of Autonomous Fixed Wing Air Vehicles," May 2009, a reference for flight dynamics, state estimation, and path planning. 8, 11, 39

[5] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing.* Upper Saddle River, NJ: Prentice Hall, 2000. 13

[6] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, vol. 92, no. 3, pp. 401–422, 1997. 15, 16

[7] R. W. Beard, personal communication, 2009. 55

[8] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls.* Lawrence, KS: Design, Analysis and Research Corporation, 2001. 56

[9] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation.* Hoboken, NJ: John Wiley and Sons, Inc., 2003. 56

[10] *TS-7800 Manual*, 1st ed., Technologic Systems Inc., Fountain Hills, AZ 85268, 2009. 91

Appendix A

Hardware Implementation

The MOTRON's basic components consist of a TS-7800 single board computer, a small LCD screen, and a numeric entry keypad. This appendix discusses the particular way in which these devices are interconnected setup, and operated.

A.1 Basic Operation of the TS-7800

The MOTRON relies primarily upon a single board computer, the TS-7800, made by Technologic Systems, Inc [10]. The TS-7800 has no video controller or keyboard interface. This is to keep the board size small and cost low. In order to interface with the device, a COM1 RS-232 connection is typically required. A null modem cable is necessary to use the COM1 Terminal plugin. An Ethernet jack is also available. Table A.1 gives the information necessary to use COM1 and the Ethernet port.

Table A.1: Interface Parameters

COM1 Serial Port		Ethernet Port	
Baud	115,200	IP Address	192.168.0.50
Data Bits	8	Username	'root'
Parity	none	Password	'hello'
Flow Control	none		
Stop Bits	1		
Jumper 2 (JP2)	On		
Username	'root'		
Password	'hello'		

On the topside of the board, shown in Fig. A.1, there are three jumpers. If jumper 1 is installed, meaning each of the jumper 1 pins are electrically connected, than the board attempts to boot from an SD Card, independent of the on-board flash memory, otherwise, the board boots from the flash memory. Two backup SD Cards have been created to safeguard the MOTRON's operation. If jumper 2 is installed,

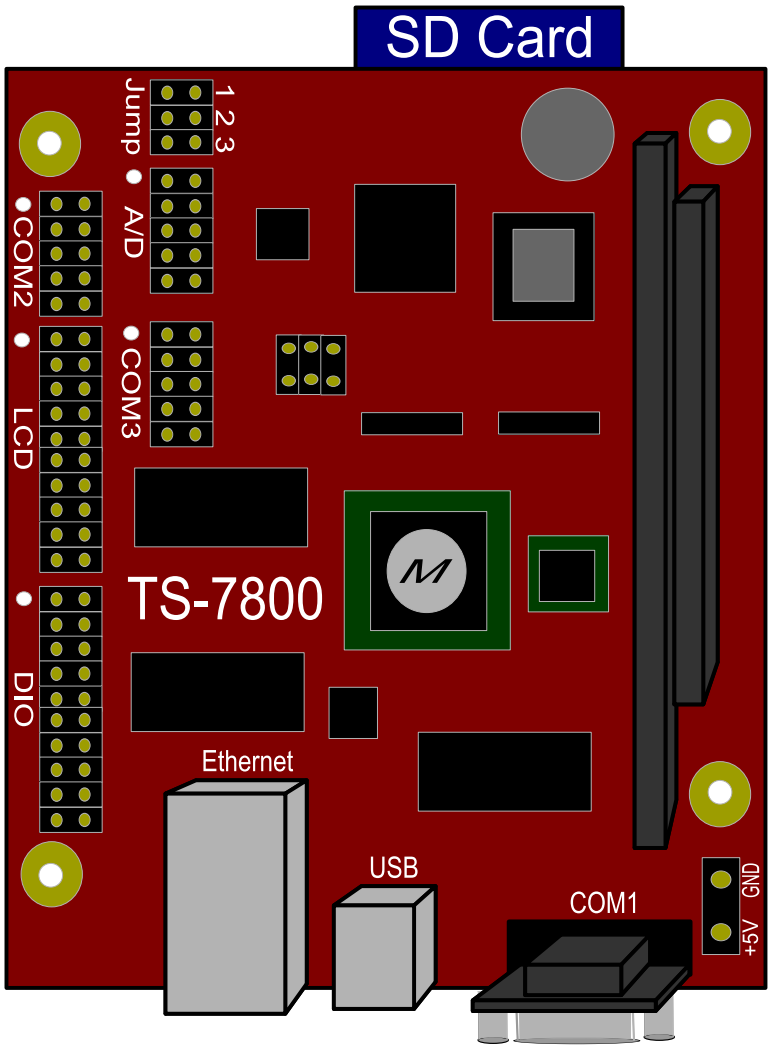

Figure A.1: The layout of the TS-7800 single board computer. Included on the TS-7800 are six separate RS-232 uarts, two USB ports, an ethernet port, an LCD port, a DIO port, an A/D, three bootup jumper options, and an SD card slot(underneath). See text for further description.

console output to COM1 is turned on, if it is not installed, console output to COM1 is turned off. If jumper 3 is installed, it causes the CPU to run at 333MHz rather than 500MHz to save power and allow it to run in hotter thermal conditions.

To turn on the TS-7800, connect a regulated 5VDC to the pins shown in the bottom left of Fig. A.1. All of the boot messages are displayed through COM1 by default. Once started, the TS-7800 boots Debian Linux from the on-board flash memory if an SD Card is not inserted. The board is set to bypass a"fast bootup" sequence and proceed directly to a full operating system bootup. To get back to the fastboot shell, touch the file "/fastboot" in the root directory of the Debian filesystem. To turn it off again, issue the command, "ln -sf /linuxrc-mtdroot /linuxrc; save". Finally, in the case of a crash, to reprogram the entire TS-7800 flash memory, boot from a backup SD Card and issue the command "createmtdroot" from the shell. This will restore all necessary files to operate the MOTRON. The MOTRON can also be operated completely from a backup SD Card.

Once booted, the TS-7800 runs a full-featured Debian Linux distribution. It includes everything necessary to run Linux and develop Linux applications. The TS-7800 also includes an on-board C/C++ compiler for developing custom applications. To compile a source file just issue the command "g++ -o outfile sourcefile.cpp". All of the MOTRON's source code is in the "/BYU" directory, including "INU.cpp", which contains the code for the MOTRON data recording program.

A.2 COM Connections

There are three COM ports on the TS-7800 which have six independent RS-232 uarts. In the "/dev" directory, they appear as ttts0, ttts1, ttyS0, ttyS1, ttts4, and ttts5. Each of these are labeled on the casing of the MOTRON. On COM1, the pins are labeled as shown in Fig. A.2, and on COM2 and COM3 the pins are labeled as in Fig. A.3. The mapping of these pins are given in Table A.2.

It is important to note that ttyS0 maps to the terminal output, and will not function properly as anything else unless jumper 2 is off. Also, ttts4 is the display/input port, and is wired directly to the LCD and keypad on the MOTRON.

Table A.2: Mapping of COM Port Pins.

COM Port	/dev	Transmit Pin	Receive Pin
COM1	ttts0	7	8
	ttts1	4	1
	ttyS0	3	2
COM2	ttyS1	3	2
COM3	ttts4	3	2
	ttts5	7	8

A special USB-Serial driver was complied to allow functionality with the NovAtel's USB features. This kernel module can be located in the kernel directory under the name "usbserial.ko". This file is different than the usual file that typically comes with Debian installations. This file has been backed up to each SD Card and does not ever need to be recross-compiled. Each time a USB device is attached, the TS-7800 automatically detects it and assigns it to a ttyUSB device name which can be found in the "/dev" directory. From that point, the USB serial device is treated just like any other serial device.

A.3 MOTRON Operating Source Code

The MOTRON's operating code, found in "/BYU/INU.cpp", is written in C++ and contains all of the serial device interfacing, NovAtel system communica-

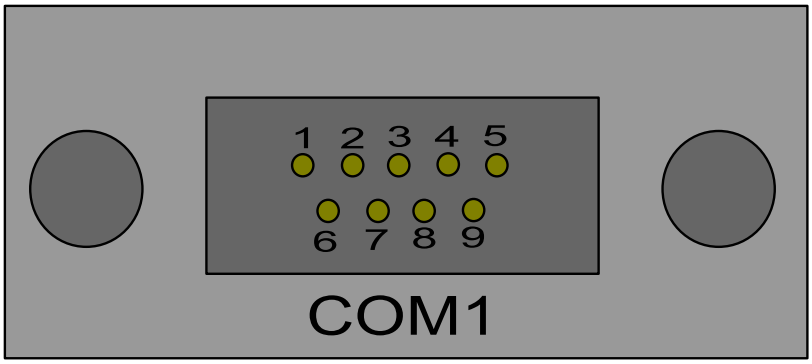

Figure A.2: The pin numbers of COM1.

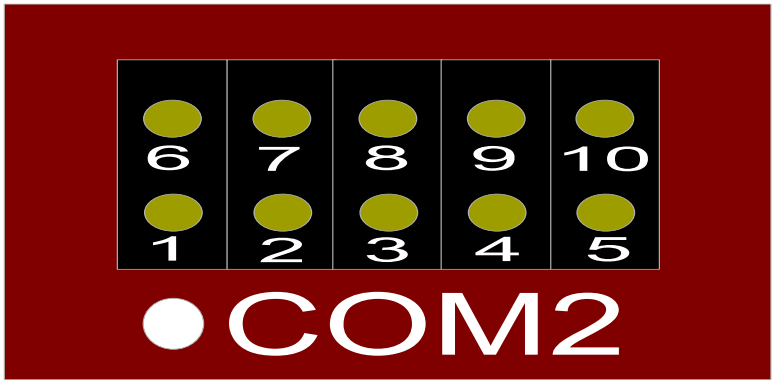

Figure A.3: The pin numbers of COM2 and COM3. Notice that the white circle indicates pin 1.

tions, and input/output handling. Serial communications were accomplished via the SerialStream package which is widely available.

A small set of LCD graphics functions are also defined. These make it much simpler to write messages to the LCD screen. A set of delays is embedded in each display function to keep the LCD display from displaying erroneous messages. Each interfacing device has an assigned Serial Port that can be found at the beginning of the main function. Each device must be interfaced within a separate thread, so the pthread package was used. Each interfacing function operates independently in order to avoid timing issues.

INU.cpp can be easily compiled on-board using the command "g++ -lserial -pthread -o INU INU.cpp". Once compiled, the script executes automatically upon system startup. In order to stop the MOTRON from immediately executing this script upon startup, edit the TS-7800's init.d script.

A.4 Additional Resources

For more information about the functionality of the TS-7800, visit Technologic System's online user manual,
http://www.embeddedarm.com/about/resource.php?item=393

For additional information regarding the LCD screen, visit
http://microcontrollershop.com/product_info.php?products_id=1974