ERROR CHARACTERIZATION OF VISION-AIDED NAVIGATION SYSTEMS

THESIS

Daniel A. Marietta, Civilian, USAF

AFIT-ENG-13-M-33

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-13-M-33

ERROR CHARACTERIZATION OF VISION-AIDED NAVIGATION SYSTEMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Daniel A. Marietta, B.S.E.E.

Civilian, USAF

March 2013

DISTRIBUTION STATEMENT A.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-13-M-33

ERROR CHARACTERIZATION OF VISION-AIDED NAVIGATION SYSTEMS

Daniel A. Marietta, B.S.E.E.
Civilian, USAF

Approved:

// signed //	12 March 2013
Lt Col K. Fisher, PhD (Chairman)	Date
// signed //	12 March 2013
Dr. C. Taylor, (Committee Member)	Date
// signed //	12 March 2013
Dr. J. Raquet, (Committee Member)	Date

Abstract

The ability to precisely know one's location in the vicinity of the Earth has been a neccessity to mankind for millenia. Today, many systems rely on the Global Positioning System (GPS) as a primary navigation aid due to both the attainable accuracy and ease of access. While GPS has arguably revolutionized the way navigation is performed today, the system possesses inherent weaknesses, spurring the search for alternative precision navigation aids. One such alternative, image-aided navigation (IAN) with stochastically constrained feature tracking has been a focus of research at the Air Force Institute of Technology's (AFIT) Advanced Navigation Technology (ANT) center. IAN systems are self contained, requiring no outside signal and thus avoid the primary weakness of satellite-based navigation systems. However, IAN systems are not yet capable of maintaining GPS-level precision without additional aiding. Particularly in systems based upon the extended Kalman filter (EKF), the navigation solution tends to be optimistic [1], degrade with time and may exhibit divergent behavior [7].

While previous research has shown that the IAN algorithm developed at the ANT center is effective at estimating the navigation states with a relative minimum of error [25], these results were based upon simulated data or small data collections (1-2 runs) and may not be indicative of real-world performance. In fact, it appears that a Monte Carlo analysis of *any* IAN system based upon a large-scale, real world data collection has yet to be presented in the literature. In addition, no work performed at the ANT center with the stochastically-constrained IAN algorithm has sufficiently addressed the issues of filter consistency and divergence. As efforts are made to improve the filter performance, it is required to first have a thorough, statistically based understanding of the filter's behavior. The goal of this work is to fulfill this requirement by characterizing the errors committed by the feature tracking, EKF-based IAN system in use at the ANT center by performing a Monte Carlo analysis of a 100 run real-world data set. The characterization of the errors

that occur in the navigation solution will serve to validate the canon of work previously performed by others at the ANT center. Additionally, it will support current and future research efforts by providing a large data set collected under controlled conditions, baseline statistics describing error behaviors, and exploration into error causes.

Acknowledgments

I would like to thank my advisor, Lt Col Fisher, and sponsor, Dr. Taylor, for their invaluable guidance, suggestions and time; the ANT center staff, particularly Mr. Kresge; and my family, for their patience and understanding as I completed my work at AFIT.

Daniel A. Marietta

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Error Characterization of Vision-Aided Navigation Systems

1 Introduction

Navigation is defined as the calculation of the position, velocity and attitude of a body with respect to time, relative to an initial starting location or origin. One focus of current research within the field is the formulation of a robust navigation solution for unmanned vehicles in environments that are devoid of a mechanism capable of absolute positioning, such as the Global Positioning System (GPS). This is due to the fact that Global Navigation Satellite Systems (GNSS), of which GPS is an example, suffer from availability restrictions when satellite signals are physically blocked in areas such as underground environments, underwater, and in urban settings. Additionally, the GPS system is susceptible to jamming and interference. This has created a push to reduce military dependence on GPS, which in turn has further increased interest in alternative navigation solutions.

1.1 Problem Definition

A coherent navigation system relies on the proper incorporation of measurements that may be obtained using one or more of a variety of sensors. Various sensors collect different types of information, and each has strengths and weaknesses in application. Some examples include:

- Laser and sonar-based sensing systems capable of extracting position information relative to a reference point or physical object.
- Cameras able to collect images of the surrounding area, which may be interpreted in a variety of ways to provide measurements of position and velocity.

- Inertial measurement units (IMUs) exploit accelerometers and gyroscopes to measure the acceleration and turn rate of the body, respectively.

- Shaft encoders can be placed on wheels of a moving vehicle to collect odometry data from which speed and relative position can be calculated.

Generally, sensor combinations are desired which can provide complementary data or counteract each other's errors. This data from these sensors must then be combined and processed to provide an estimation of the navigation states. This can be done either in real-time or after data collection has occurred. The typical method of combining the sensor measurements has been through the implementation of the Kalman filter or one of its derivatives. Specifically, the Extended Kalman filter (EKF) is often used, as it allows for non-linearities in system dynamics and measurement relationships by linearizing these functions about nominal trajectory points. However, the EKF suffers from problems in implementation; in particular, errors can grow unbounded over time, leading to the divergence of the filter solution.

The EKF *estimates* the states of interest by propagating the mean and covariance of these estimates in time, conditioned on the time history of measurements. This provides all information required to fully describe the state estimates, since they are assumed to be Gaussian in nature. In addition, as the true state $\mathbf{x}(t_i)$ and state estimates $\hat{\mathbf{x}}(t_i)$ can be shown to be jointly Gaussian, the error committed by the filter

$$\boldsymbol{\epsilon}(t_i) \triangleq \mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i) \tag{1.1}$$

is also a Gaussian random process [14]. In practice however, this description does not hold true. During the course of any given implementation, the EKF may converge, or it may not. When the ensemble statistics are observed over many realizations, it becomes clear that the ideal description of the error committed by the filter as being a Gaussian random process does not mesh with reality. In general, it is known that unmodeled errors

may be introduced into the EKF through the use of the series expansion in measurement and state prediction [2], resulting in an inconsistent filter.

Based upon the information provided in the preceding paragraphs, it is evident that computing a proper and accurate navigation solution is a beast that is difficult to tame, particularly if the scheme in question relies on the EKF. There are a great number of issues that must be explored and well understood if a proposed navigation algorithm is to be implemented. One such algorithm, the stochastically constrained Image Aided Extended Kalman Filter (IAEKF) developed in [25] is a current focus of research at the Air Force Institute of Technology's (AFIT) Advanced Navigation Technology (ANT) Center. The algorithm relies on inertial and image measurements, combined with past estimates via an EKF, to estimate a navigation solution. The purpose of this work is to leverage this algorithm to explore issues pertinent to Image-Aided Navigation (IAN). Specifically, the algorithm is used to process a collection of 100 data sets, which are then analyzed using Monte Carlo analysis techniques in an effort to characterize the errors committed by the filter. The algorithm employed is discussed in detail in Section 2.10.1, and the nature of the data is addressed in Section 3.2. Filter computed mean and covariance estimates for the navigation states will be statistically described and compared with truth sources. Divergent runs are explored separately, and causes of divergence are discussed. The description of the errors committed and how they compare with ideal or expected values, as well as the description of the rate and symptoms of divergence, form a full characterization of filter errors. This error characterization is the primary goal of this work.

1.2 Research Contributions

The primary contribution of this research is the development of baseline statistics describing IAN errors, which will serve as a point of reference for other works in IAN in general, and more specifically, to those working with the stochastically constrained IAN

algorithm considered herein. Expected rates of filter divergence, the causes of divergence, and possible solutions are also explored. As a secondary benefit, the large size of the data pool and relative uniformity of the collection environment over each run means that this data set can be easily used in other IAN research endeavors; no prior effort has produced a real-world, 100 run data collection. In particular, this data pool, and the analysis contained in this work, are meant to provide a background for further investigation into outlier aware filtering, presented in [20].

1.3 Outline

The remainder of this work is organized in the following manner: Chapter 2 provides an overview of the necessary mathematical background required to understand the performance of the IAN algorithm, and techniques which are used in analysis of the results. Chapter 3 describes the data collection process, the physical hardware of the collection platform and all processing that is performed on the data before it proceeds through the navigation filter. Chapter 4 encompasses the contributions of this work; the collected data is described, and the results of the navigation filter are thoroughly analyzed and discussed. Finally, Chapter 5 provides a summary of the work by encapsulating the conclusions that are drawn from the results provided in Chapter 4. Additionally, future research topics and suggested improvements that should be considered if research similar to this is undertaken at a later date are provided.

2 Background

This chapter presents concepts central to image-aided inertial navigation and statistical analysis of the same. The mathematical notation that is used throughout this work is presented first in Section 2.1. Sections 2.2 through 2.9 each addresses one of the various background topics that collectively underpin IAN. A literature review that summarizes publications which have had a significant impact on enabling or motivating this work is provided in Section 2.10. Finally, a summary of the chapter contents is given in Section 2.11.

2.1 Mathematical Notation

The following conventions will be applied throughout the document:

Scalars: Scalar quantities are denoted by either upper or lower case characters in italics, as in b or B.

Vectors: Vectors are denoted as boldface lower case characters, as in $\mathbf{a}$ or $\mathbf{x}$. All vectors are to be assumed column vectors unless otherwise stated.

Vector Components: Individual vector components that represent scalar values along a specific axis are denoted as the vector with a subscript representing the corresponding axis, as in $\mathbf{a}_x$ or $\mathbf{k}_z$.

Homogeneous Vectors: A homogeneous vector is constructed by adding an additional component to a standard vector, equal to 1. These are represented as vectors with an underscore, as in $\underline{\mathbf{a}}$.

Unit Vectors: A unit vector for an arbitrary vector $\mathbf{v}$ is obtained by dividing the vector components by the magnitude of the vector length, $\frac{\mathbf{v}}{\|\mathbf{v}\|}$. Within this document, unit vectors are marked with a tilde, as in $\widetilde{\mathbf{v}}$.

Matrices: Matrices are denoted as upper case, boldface characters, as in $\mathbf{A}$ or $\mathbf{\Omega}$. Specific elements in a matrix are designated by subscripts i.e., the element of matrix $\mathbf{A}$ in the ith row and jth column is shown as $\mathbf{A}_{ij}$.

Matrix Transpose: The transpose of a matrix or vector is represented by a superscript upper case T, as in $\mathbf{A}^{\mathrm{T}}$ or $\mathbf{a}^{\mathrm{T}}$.

Estimates: An estimate of a variable is designated by the hat character, as in $\hat{\mathbf{x}}$.

Reference Frames: A superscript letter will be used to denote a vector expressed in a given reference frame, as in $\mathbf{a}^{\mathrm{c}}$.

Direction Cosine Matrices: The direction cosine matrix that rotates a vector from frame a to frame b is shown as $\mathbf{C}_a^b$.

Transformation Matrices: Similar to a direction cosine matrix, a transformation matrix rotates a vector as does a direction cosine matrix, but it also performs a translation on the vector. These will be expressed in the form $\mathbf{T}_a^b$.

Mapping Function: When a function is applied to a set of coordinates as to map them to a different space than that in which they were originally defined (such as a mapping from $\mathbb{R}^3$ space to $\mathbb{R}^2$), an arrow will be used with the initial and final space denoted on top, separated by a colon, as in $\overset{\mathbb{R}^3:\mathbb{R}^2}{\rightarrow}$.

Relative Motion: Two subscript letters will be used to represent value that relates different reference frames e.g., ω_{ab} is the rotation rate vector between frame a and frame b .

A Priori and A Posteriori Estimates: The *a priori* and *a posteriori* estimates within a Kalman filter will be designated by superscript minus and plus characters respectively, as in $\hat{\mathbf{x}}(t^-)$ and $\hat{\mathbf{x}}(t^+)$.

Any departures in the text from the above notational conventions will be expressly identified as such.

2.2 Reference Frames

A reference frame is an oriented vector space in which an object can be located and relative movement described using a defined coordinate system. The following reference frames, with definitions taken from [21] and [25], are used within this research:

True inertial frame (*I*-frame): The *I*-frame is a theoretical reference frame with no pre-defined orientation or origin. It is defined as a non-accelerating, non-rotating orthonormal basis, affixed to an arbitrary origin in $\mathbb{R}^3$, in which Newton's laws apply.

Earth-centered Earth-fixed frame (*e*-frame): The *e*-frame is a three-dimensional orthonormal frame whose origin lies at the center of mass of the Earth. The *z*-axis extends through the International Earth Rotation Service (IERS) Reference Pole, the *x*-axis extends from the origin through the intersection of the Prime Meridian and the Equator, and the *y*-axis is defined as to complete the right-handed, orthonormal set. The frame rotates about the *z*-axis. This frame is the same as the WGS-84 Coordinate System [4], and is shown in Figure 2.1.

Earth-centered inertial frame (*i*-frame): The Earth-centered inertial frame has an origin and *z*-axis defined as they are for the *e*-frame, but the *x*-axis points out from the origin to the first star in Aries. The *y*-axis completes orthonormality. The star to which the *x*-axis points is considered fixed, and thus this frame does not rotate. It can therefore be used as an inertial reference frame for the purpose of navigation on the Earth.

Navigation frame (*n*-frame): Depicted in Figure 2.1, the *n*-frame is a local-level frame usable when the curvature of the Earth over a given distance is considered negligible. The origin is chosen to be at a fixed location in the local area. Coordinates are then defined with respect to this point, assuming flat-earth geometry. The *x, y,* and *z*

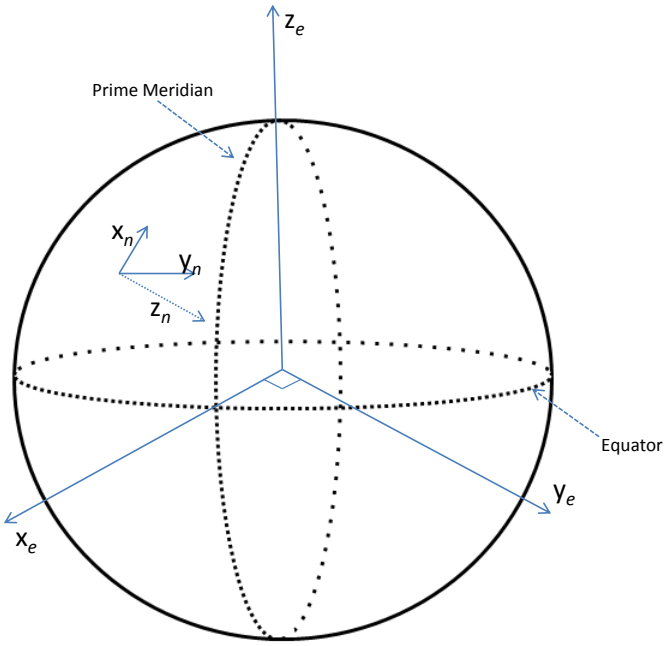

Figure 2.1: The *e* and *n*-frames. The *n*-frame is only valid over a region in which the curvature of the Earth has been deemed negligible.

coordinates respectively reference the North, East,and Down (NED) directions with respect to the origin. Down is defined here by the local gravity vector (the direction a hanging plumb bob would point).

Body frame (*b*-frame): The *b*-frame is used to orient a point in reference to a vehicle. The origin is typically chosen to be either co-located with a sensor location on the body, or at the body's center of mass. The *x*, *y*, and *z* axes are defined as going through the nose of the vehicle, out of the right side, and downwards through the body, respectively. These are aligned with the roll, pitch, and yaw axes of the vehicle. The *b*-frame is illustrated in Figure 2.2.

Camera frame (*c*-frame): The *c*-frame is defined with an origin that coincides with the optical center of the camera, *x* and *y* axes that point up and to the right, respectively, and a *z*-axis that points out of the lens, as shown in Figure 2.3. The *x* and *y* axes are

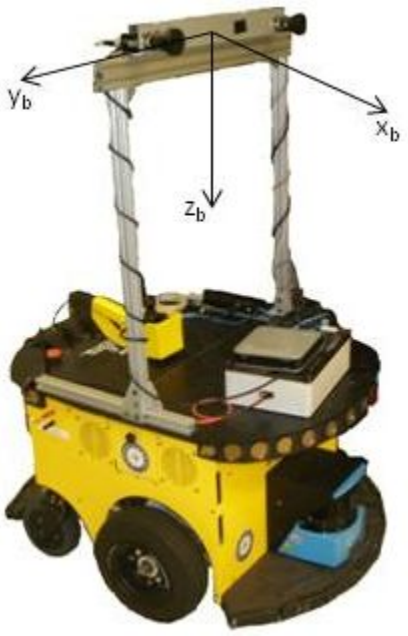

Figure 2.2: The *b*-frame. The origin of the body frame is attached to the center of the MIDG II IMU, which is mounted on a frame attached to the robot. The x_b-axis is directed out of the front of the robot, the y_b-axis to the right-hand side, and the z_b-axis points down out of the bottom of the robot. The three axes form an orthogonal triad.

parallel with the image plane of the camera, while the *z*-axis lies perpendicular to the image plane.

Image frame (*pix*-frame): The *pix*-frame is used to describe pixel locations within digital images. It is an orthonormal basis in $\mathbb{R}^2$ with the origin defined as the upper-left corner of the image. Popular conventions for the coordinates of the upper-left pixel include denoting it as location [0,0] or [1,1]. The second option will be used within this work to adhere to the indexing conventions found in MATLAB®.

2.3 Coordinate Transformations

A common procedure is representing a vector quantity in a reference frame other than that in which it was originally defined. When two reference frames are not co-located, it is necessary to quantify the differences in location and orientation between

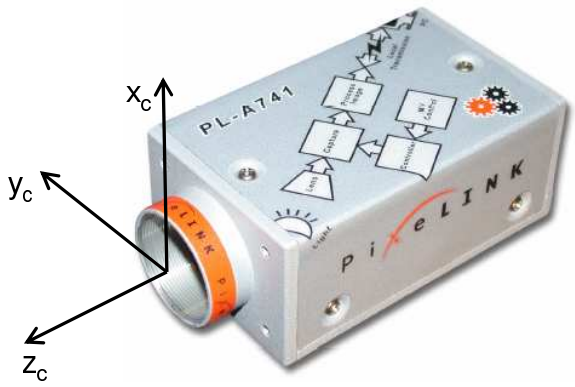

Figure 2.3: Camera Frame. From [25].

the two. This can be done through the use of direction cosine matrices and translation vectors. Additionally, these two expressions can be combined into a transformation matrix, which encompasses the effects of both. The information presented in this section draws on descriptions of the same topic found in [9] and [21].

2.3.1 Direction Cosine Matrices. The representation used in this research to relate the orientations of two reference frames is the Direction Cosine Matrix (DCM). The following overview of the DCM equations is adapted from [21].

The process of aligning a given reference frame *a* with another frame *b* involves the rotation of frame *a* about three axes, and thus three angles of rotation must be defined. These angles are known as Euler angles, and are applied in the following manner. First, frame *a* is rotated through angle Ψ about its *z*-axis, resulting in intermediate frame *a'* . Next, *a'* is rotated through angle Θ about its own *y*-axis, forming frame *a"*. Finally, *a"* is rotated through angle Φ about its *x*-axis. A DCM can be used to describe each of these rotations.

The matrices $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ apply rotations about the z, y, and x axes, and are respectively defined as

$$\mathbf{C}_1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\mathbf{C}_2 = \begin{bmatrix} \cos\Theta & 0 & -\sin\Theta \\ 0 & 1 & 0 \\ \sin\Theta & 0 & \cos\Theta \end{bmatrix} \tag{2.2}$$

$$\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{bmatrix} \tag{2.3}$$

The DCM that transforms a vector from the reference frame a to frame b is then

$$\mathbf{C}_a^b = \mathbf{C}_3\mathbf{C}_2\mathbf{C}_1 \tag{2.4}$$

The DCM for transformation of frame b to frame a is $\mathbf{C}_b^a = (\mathbf{C}_a^b)^{\mathrm{T}}$. A vector $\mathbf{p}$, originally described with respect to reference frame b, is described in frame a coordinates by pre-multiplying the vector with the appropriate DCM, as in

$$\mathbf{p}^a = \mathbf{C}_b^a\mathbf{p}^b \tag{2.5}$$

DCMs also possess the following properties [9]:

1. The determinant of a DCM is always equal to 1 if both frames are comprised of right-handed, orthonormal bases.
2. The inverse of a DCM always exists.
3. The inverse of a DCM is equal to its transpose.
4. $\mathbf{C}_a^c = \mathbf{C}_b^c\mathbf{C}_a^b$

As the attitude of rotating frame b changes with time in relation to a reference frame a, it will become necessary to update the DCM relating the two. Assuming that the change in orientation of frame b over a time step δt is small, a DCM relating frame b at time t to frame b at time $t + \delta t$, $\mathbf{A}(t)$, can be constructed

$$\mathbf{A}(t) = [\mathbf{I} + \delta\mathbf{\Psi}] \tag{2.6}$$

where $\mathbf{I}$ is a 3x3 identity matrix and

$$\delta\mathbf{\Psi} = \begin{bmatrix} 0 & -\delta\psi & \delta\Theta \\ \delta\psi & 0 & -\delta\Phi \\ -\delta\Theta & \delta\Phi & 0 \end{bmatrix} \tag{2.7}$$

The values $\delta\psi$, $\delta\Theta$, and $\delta\Phi$ are the angles through which the b-frame has rotated about its z, y , and x axes, respectively, over the period δt. The updated DCM can then be calculated as

$$\mathbf{C}_b^a(t + \delta t) = \mathbf{C}_b^a(t)\mathbf{A}(t) \tag{2.8}$$

The time rate of change of the DCM $\mathbf{C}_b^a$ is given as

$$\dot{\mathbf{C}}_b^a = \lim_{\delta t \to 0} \frac{\mathbf{C}_b^a(t + \delta t) - \mathbf{C}_b^a(t)}{\delta t} \tag{2.9}$$

As δt approaches 0, $\delta\mathbf{\Psi}/\delta t$ is the skew-symmetric matrix of the angular rate vector $\boldsymbol{\omega}_{ab}^b = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^{\mathrm{T}}$. In other words,

$$\lim_{\delta t \to 0} \frac{\delta\mathbf{\Psi}}{\delta t} = \mathbf{\Omega}_{ab}^b \tag{2.10}$$

where

$$\mathbf{\Omega}_{ab}^b = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.11}$$

Thus the time rate of change of the DCM $\mathbf{C}_b^a$ may be expressed as

$$\dot{\mathbf{C}}_b^a = \mathbf{C}_b^a\mathbf{\Omega}_{ab}^b \tag{2.12}$$

2.3.2 Translation Vectors. As stated at the beginning of this section, reference frames may differ not only in their orientation with respect to one another, but also in the relative positions of their origins. In this case, the transformation of a vector represented in a-frame coordinates to the b-frame is accomplished using

$$\mathbf{p}^b = \mathbf{C}_a^b(\mathbf{p}^a - \mathbf{p}_{ab}^a) \tag{2.13}$$

where $\mathbf{p}_{ab}^a$ is the vector pointing from the origin of the a-frame to the origin of the b-frame, expressed in the a-frame coordinate system.

2.3.3 Transformation Matrices. A transformation matrix combines the effects of a DCM and translation vector, and is an alternate representation of Equation 2.13. The translation vector and DCM relating frame a to frame b are used to create an augmented matrix of the form

$$\mathbf{T}_a^b = \begin{bmatrix} \mathbf{C}_a^b & | & -\mathbf{p}_{ab}^b \\ -- & & -- \\ 0_{1x3} & | & 1 \end{bmatrix} \tag{2.14}$$

The next step is to obtain the homogeneous form of the vector to be transformed from a coordinates to b coordinates. This operation entails augmenting the vector $\mathbf{v}^a$ with another component equal to 1, resulting in

$$\underline{\mathbf{v}}^a = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}^{\mathrm{T}} \tag{2.15}$$

With these two pieces in place, the depiction of vector $\underline{\mathbf{v}}^a$ in b-frame coordinates can be found by pre-multiplying it with the transformation matrix

$$\underline{\mathbf{v}}^b = \mathbf{T}_a^b \underline{\mathbf{v}}^a \tag{2.16}$$

2.4 Inertial Navigation

Inertial navigation is defined as using data provided by inertial sensors to compute the position, velocity, and attitude of a vehicle over time [21]. If the sensors are affixed to

the frame of the body being monitored, this is referred to as strapdown inertial navigation. In this section, an overview of strapdown inertial navigation concepts is presented, including sources of measurements and derivations of equations for navigation in pertinent reference frames. Additional information can be found in both [21] and [25].

2.4.1 Acceleration and Specific Force. Strapdown inertial navigation is concerned with computing the position, velocity, acceleration and attitude of a vehicle at some point in time using measurements taken from inertial sensors mounted on the vehicle, namely accelerometers and gyroscopes. These are typically arranged in orthogonal triads and sense acceleration in and angular rate about three axes.

Accelerometers do not measure inertial acceleration, but instead provide a measurement of specific force [21]. Specific force is the difference between the sensed acceleration with respect to an inertial reference frame and the acceleration due to gravity or mass attraction. Therefore, specific force can be defined as [21]

$$\mathbf{f} = \ddot{\mathbf{p}} - \mathbf{G} \tag{2.17}$$

where $\ddot{\mathbf{p}}$ is the acceleration vector, $\mathbf{f}$ is the specific force vector, and $\mathbf{G}$ is gravity due to mass attraction.

The gyroscope, commonly referred to as a gyro, is used to measure the angular rate of a body relative to inertial space, $\boldsymbol{\omega}_{ib}^{b}$. The measurement provided by the gyro can be expressed in skew-symmetric form as

$$\boldsymbol{\Omega}_{ab}^{b} = \mathbf{C}_{a}^{b}\dot{\mathbf{C}}_{b}^{a} \tag{2.18}$$

by rearranging Equation 2.12. Information from the accelerometers and gyros can be used in conjunction with the strapdown mechanization equations to determine the trajectory of a body with respect to time. The relevant mechanizations, in the *e* and *n* frames, are presented below. Derivations can be found in both [21] and [25].

2.4.2 Inertial Navigation System Equations. Before presenting the Inertial Navigation System (INS) mechanization equations, it is first necessary to define the *local gravity vector* at point **p** , which is the difference between the gravitational force due to mass attraction at a given point and the centripetal force experienced by the accelerometer due to the rotation of the Earth [21]

$$\mathbf{g} = \mathbf{G} - \boldsymbol{\omega}_{ie} \times [\boldsymbol{\omega}_{ie} \times \mathbf{p}] \tag{2.19}$$

where **g** is the local gravity vector, **G** is the gravitational acceleration due to mass attraction, and $\boldsymbol{\omega}_{ie} \times [\boldsymbol{\omega}_{ie} \times \mathbf{p}]$ is the acceleration the object experiences as a result of the centripetal force applied due to the rotation of the Earth $\boldsymbol{\omega}_{ie}$. The total acceleration experienced by a body is a function of the specific force acting upon the body, the Coriolis acceleration due to the rotation of the Earth, and the acceleration due to the local gravity vector. The acceleration of a body navigating in the vicinity of the Earth, expressed in *e*-frame coordinates is:

$$\ddot{\mathbf{p}}^e = \mathbf{C}_b^e \mathbf{f}^b - 2\boldsymbol{\omega}_{ie}^e \times \mathbf{v}^e + \mathbf{g}^e \tag{2.20}$$

However, navigation during the course of this work will be conducted using a wheeled robot, affixed to the surface of the Earth. This platform rotates with respect to the *i*-frame at the same rate as the Earth itself. This allows for the effect of Coriolis acceleration to be neglected, and Equation 2.20 reduces to:

$$\ddot{\mathbf{p}}^e = \mathbf{C}_b^e \mathbf{f}^b + \mathbf{g}^e \tag{2.21}$$

An expression for position in the *n*-frame can be derived from a vector translation in *e*-frame coordinates relating the origins of the *e* and *n* frames, converted for representation in the *n*-frame:

$$\mathbf{p}^n = \mathbf{C}_e^n [\mathbf{p}^e - \mathbf{p}_{n_0}^e] \tag{2.22}$$

Twice taking the derivative of Equation 2.22 and rearranging yields an expression for the acceleration of the body in the n-frame:

$$\ddot{\mathbf{p}}^n = \mathbf{f}^n - 2\mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e\dot{\mathbf{p}}^n - \mathbf{C}_e^n(\boldsymbol{\Omega}_{ie}^e)^2[\mathbf{p}_0^e + \mathbf{C}_n^e\mathbf{p}^n] + \mathbf{g}^n \tag{2.23}$$

The corresponding equation for the attitude estimate is

$$\dot{\mathbf{C}}_b^n = \mathbf{C}_b^n\boldsymbol{\Omega}_{ib}^b - \mathbf{C}_e^n\boldsymbol{\Omega}_{ie}^e\mathbf{C}_n^e\mathbf{C}_b^n \tag{2.24}$$

where $\mathbf{C}_y^z$ is the DCM from the generic y-frame to the z-frame, and $\boldsymbol{\Omega}_{yz}^z$ is the y-frame to z-frame angular rate expressed in the z-frame [25].

2.5 Kalman Filtering

2.5.1 Linear Kalman Filter. The Kalman filter is an optimal, recursive data processing algorithm [14]. It combines measurements and knowledge of the system to provide state estimates with a minimum of error. Generally, the Kalman filter is provided with a linear model of the system in question, with a continuous-time representation

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.25}$$

where $\mathbf{x}$ is vector of the system states, $\mathbf{u}$ vector of control inputs, and $\mathbf{w}$ is a vector of white, Gaussian noise processes. The noise process $\mathbf{w}$ is of zero mean with covariance kernel

$$E\{\mathbf{w}(t)\mathbf{w}(t+\tau)\} = \mathbf{Q}\delta(\tau) \tag{2.26}$$

where E is the expectation operator and $\delta(\tau)$ is the Dirac delta function. Further, the $\mathbf{F}$(t), $\mathbf{B}$(t), and $\mathbf{G}$(t) terms seen in Equation 2.25 are known as the system dynamics, input, and noise transformation matrices, respectively. If the system is time-invariant, the time indices on $\mathbf{F}$, $\mathbf{B}$, and $\mathbf{G}$ may be dropped, resulting in

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{F}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{2.27}$$

The continuous-time equation is not well-suited to digital implementation, and thus a discretized form is required. From [14], the discrete form of Equation 2.25 is given as

$$\mathbf{x}(t_{i+1}) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{x}(t_i) + \mathbf{B}_\mathrm{d}(t_i)\mathbf{u}(t_i) + \mathbf{G}_\mathrm{d}(t_i)\mathbf{w}_\mathrm{d}(t_i) \tag{2.28}$$

The discretized version of the control inputs $\mathbf{u}(t)$ are obtained by sampling the continuous time control inputs at appropriate intervals. The remainder of the terms appearing in Equation 2.27, $\mathbf{F}$, $\mathbf{B}$, $\mathbf{G}$, and $\mathbf{w}(t)$, must each be converted into a discrete time representation. The methods used for performing this process follow from [14].

The discrete state transition matrix $\mathbf{\Phi}(t - t_0)$ can be calculated from the $\mathbf{F}$ matrix as

$$\mathbf{\Phi}(t - t_0) = e^{\mathbf{F}(t-t_0)} \tag{2.29}$$

Further, from the general solution to Equation 2.27, (with the term $\beta(\tau)$ representing a Brownian motion source of diffusion q(t)),

$$\mathbf{x}(t) = \mathbf{\Phi}(t - t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{\Phi}(t - \tau)\mathbf{B}\mathbf{u}(\tau)\delta\tau + \int_{t_0}^{t} \mathbf{\Phi}(t - \tau)\mathbf{G}\delta\beta(\tau) \tag{2.30}$$

which is found through the use of linear algebra techniques, the discretized $\mathbf{B}$ matrix $\mathbf{B}_d$ can be solved for using

$$\mathbf{B}_d = \mathbf{F}^{-1}(e^{\mathbf{F}(t-t_0)} - \mathbf{I})\mathbf{B} \tag{2.31}$$

The stochastic process $\mathbf{w}_\mathrm{d}$ is zero-mean with covariance kernel

$$E\{\mathbf{w}_\mathrm{d}(t_i)\mathbf{w}_\mathrm{d}^\mathrm{T}(t_j)\} = \mathbf{Q}_\mathrm{d}\delta_{ij} \tag{2.32}$$

Calculation of $\mathbf{Q}_d$, may be accomplished using methods outlined in [22]. In this research however, a second-order Taylor series approximation

$$\mathbf{Q}_d \approx \frac{1}{2}(\mathbf{\Phi}(t - t_0)\mathbf{Q}_t\mathbf{\Phi}^\mathrm{T}(t - t_0) + \mathbf{Q}_t)(t - t_0) \tag{2.33}$$

is used, as presented in [14]. This approximation is suitable as the integration time is small; additional benefit is gained due to the approximation being faster to compute than

the method presented in [22]. The calculation of $\mathbf{Q}_{\mathrm{d}}(t_i)$ incorporates the effects of the continuous time noise input matrix $\mathbf{G}$. Thus $\mathbf{G}_{\mathrm{d}}$, the discrete time version of $\mathbf{G}$, becomes an $m \times n$ identity matrix, where m is the length of the $\mathbf{w}$, and n is the length of the state vector $\mathbf{x}$. This reduces Equation 2.28 to

$$\mathbf{x}(t_{i+1}) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{x}(t_i) + \mathbf{B}_{\mathrm{d}}(t_i)\mathbf{u}(t_i) + \mathbf{w}_{\mathrm{d}}(t_i) \tag{2.34}$$

Given the discrete form of the system model shown in Equation 2.34, the next matter to address is the incorporation of discrete-time measurements. These measurements are brought into the system according to the measurement model equation

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.35}$$

where $\mathbf{z}(t_i)$ are the measurements available at time t_i, $\mathbf{x}(t_i)$ is the state vector at time t_i, $\mathbf{H}$ is the measurement matrix, and $\mathbf{v}(t_i)$ is a vector of white, discrete Gaussian noise processes described by a mean and covariance kernel of

$$E\{\mathbf{v}(t_i)\} = \mathbf{0} \tag{2.36}$$

and

$$E\{\mathbf{v}(t_i)\mathbf{v}^{\mathrm{T}}(t_j)\} = \mathbf{R}\delta_{ij} \tag{2.37}$$

with δ_{ij} being the Kronecker delta function. In addition, $\mathbf{w}_{\mathrm{d}}(t)$ and $\mathbf{v}(t)$ are assumed independent of each other, and thus uncorrelated, since they are also assumed to be Gaussian [14].

With an initial state estimate vector and covariance matrix, $\hat{\mathbf{x}}(0)$ and $\mathbf{P}_{xx}(0)$ respectively, and $\hat{\mathbf{x}}(0)$ assumed to be Gaussian and independent of the noise sources within the system, the Kalman filter algorithm equations can be presented. The algorithm is divided into two processes: propagation and update. Propagation occurs from one time step to the next in the absence of new measurement information and is characterized by a

growth in state uncertainty with time. The update process incorporates new measurements and optimally combines this information with the previous state.

The propagation equations are given as

$$\hat{\mathbf{x}}(t_{i+1}^-) = \mathbf{\Phi}\hat{\mathbf{x}}(t_i^+) + \mathbf{B}_d\mathbf{u}(t_i) \tag{2.38}$$

$$\mathbf{P}_{xx}(\mathrm{t}_{i+1}^-) = \mathbf{\Phi}\mathbf{P}_{xx}(\mathrm{t}_i^+)\mathbf{\Phi}^{\mathrm{T}} + \mathbf{Q}_d \tag{2.39}$$

Equation 2.38 propagates the a posteriori state estimate at time t_i to the a priori estimate at time t_{i+1} in absence of a new measurement, incorporating any command input that may exist. Equation 2.39 propagates the a posteriori estimate uncertainty at time t_i to the a priori uncertainty at time t_{i+1}.

When a new measurement becomes available at time t_i, this additional information must be incorporated by the algorithm. The equations used to accomplish this task are

$$\mathbf{K}(\mathrm{t}_i) = \mathbf{P}_{xx}(\mathrm{t}_i^-)\mathbf{H}^{\mathrm{T}}[\mathbf{H}\mathbf{P}_{xx}(\mathrm{t}_i^-)\mathbf{H}^{\mathrm{T}} + \mathbf{R}]^{-1} \tag{2.40}$$

$$\hat{\mathbf{x}}(\mathrm{t}_i^+) = \hat{\mathbf{x}}(\mathrm{t}_i^-) + \mathbf{K}(\mathrm{t}_i)[\mathbf{z}(\mathrm{t}_i) - \mathbf{H}\hat{\mathbf{x}}(\mathrm{t}_i^-)] \tag{2.41}$$

$$\mathbf{P}_{xx}(\mathrm{t}_i^+) = \mathbf{P}_{xx}(\mathrm{t}_i^-) - \mathbf{K}(\mathrm{t}_i)\mathbf{H}\mathbf{P}_{xx}(\mathrm{t}_i^-) \tag{2.42}$$

Equation 2.40 is known as the Kalman gain equation. The Kalman gain, $\mathbf{K}(t_i)$, is a parameter used within Equations 2.41 and 2.42 to assign a level of preference to the previous estimate and the new measurements. For example, if it has been a relatively long time since the last successful update, such that the propagated state uncertainty $\mathbf{P}_{xx}(t_i^-)$ has grown large as compared to the measurement uncertainty $\mathbf{R}$, preference will be given to the measurement such that the mean squared error is minimized.

2.5.2 Extended Kalman Filter. The Extended Kalman Filter (EKF) is a variation of the linear Kalman filter that may be used when non-linear state dynamics or measurement models are present and cannot be neglected [15]. Essentially, the system is linearized about the current state estimate, propagated until a measurement is available,

and the state estimate and covariance is updated. A new estimate $\hat{\mathbf{x}}(t_i^+)$ is produced, and the entire system is then re-linearized about this estimate. The process repeats for all applicable time t, re-linearizing about each successive state estimate.

Given a system with non-linear dynamics and additive noise described as

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] + \mathbf{G}(t)\mathbf{w}(t) \tag{2.43}$$

where $\mathbf{w}(t)$ is a zero-mean white Gaussian noise process with covariance kernel

$$E\{\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(t+\tau)\} = \mathbf{Q}(t)\delta(\tau) \tag{2.44}$$

The discrete-time measurements associated with this system are related to the true state $\mathbf{x}(t_i)$ through

$$\mathbf{z}(t_i) = \mathbf{h}[\mathbf{x}(t_i), t_i] + \mathbf{v}(t_i) \tag{2.45}$$

where the measurement function $\mathbf{h}[\cdot]$ may contain non-linearities, and $\mathbf{v}(t_i)$ is zero-mean additive, white Gaussian noise with covariance kernel

$$E\{\mathbf{v}(t_i)\mathbf{v}^{\mathrm{T}}(t_j)\} = \mathbf{R}\delta_{ij} \tag{2.46}$$

Assume that the initial state $\mathbf{x}(t_0)$ is a Gaussian random vector with the following mean

$$E\{\mathbf{x}(t_0)\} = \hat{\mathbf{x}}_0 \tag{2.47}$$

and covariance

$$E\{[\mathbf{x}(t_0) - \hat{\mathbf{x}}_0][\mathbf{x}(t_0) - \hat{\mathbf{x}}_0]^{\mathrm{T}}\} = \mathbf{P}_0 \tag{2.48}$$

A nominal trajectory can be generated over the time interval $[t_i, t_{i+1})$, representing the best available estimate of the state during this period by integrating the system dynamics equation

$$\dot{\mathbf{x}}_n(t/t_i) = \mathbf{f}[\mathbf{x}_n(t/t_i), \mathbf{u}(t), t] \tag{2.49}$$

where the notation $x_n(t/t_i)$ denotes the the trajectory at time t conditioned on measurements received up to time t_i, and the initial condition for the trajectory is based on the last state estimate $\mathbf{x}_n(t_i/t_i) = \hat{\mathbf{x}}(t_i^+)$.

A perturbation model for the states $\delta\mathbf{x}(t)$ may be defined as

$$\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_{\mathrm{n}}(t), \tag{2.50}$$

and the perturbed state $\delta\mathbf{x}(t)$ can be found at timestep t by taking the Jacobian of the dynamics model and expanding about the current state estimate $\mathbf{x}_n(t)$, resulting in

$$[\dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_n(t)] \approx \left.\frac{\partial \mathbf{f}[\mathbf{x}, \mathbf{u}(t), t]}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_n(t)} [\mathbf{x}(t) - \mathbf{x}_{\mathrm{n}}(t)] + \mathbf{G}(t)\mathbf{w}(t) \tag{2.51}$$

Equation 2.51 is analogous to a Taylor series expansion about $\mathbf{x}_n(t)$. Terms higher than first order are dropped, amounting to a linearization of $\mathbf{f}$. The relationship in Equation 2.51 is therefore shown as an approximation rather than an equivalency. This relationship is appropriate only if the perturbations from the nominal trajectory are small enough for the higher order terms to be neglected [15]. To simplify notation, denote the linearized version of $\mathbf{f}$ as $\mathbf{F}$, i.e.,

$$\mathbf{F}[t; \mathbf{x}_n(t)] \triangleq \left.\frac{\partial \mathbf{f}[\mathbf{x}, \mathbf{u}(t), t]}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_n(t)} \tag{2.52}$$

In a similar manner, if the measurement equation $\mathbf{h}$ contains non-linearities, it is linearized through the same Jacobian process applied to $\mathbf{f}$

$$\mathbf{H}[t_i; \mathbf{x}_n(t_i)] \triangleq \left.\frac{\partial \mathbf{h}[\mathbf{x}, t_i]}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_n(t_i)} \tag{2.53}$$

The linearized measurement model is denoted as $\mathbf{H}$, analogous to the notation applied to linearized $\mathbf{f}$ in Equation 2.52. Evaluation of Equation 2.45, with $\mathbf{H}$ replacing the non-linear $\mathbf{h}$, results in measurement perturbations $\delta\mathbf{z}(t_i)$, where

$$\delta\mathbf{z}(t_i) = \mathbf{H}[t_i; \mathbf{x}_n(t_i)]\delta\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.54}$$

In a typical application, the state and covariance matrix will be propagated in discrete time. Assuming that this is the case, the state transition matrix $\mathbf{\Phi}$ can be found through the normal means, namely

$$\mathbf{\Phi}(t) = e^{\mathbf{F}(t)\Delta t} \tag{2.55}$$

Using the states $\delta\mathbf{x}(t_i)$ and measurements $\delta\mathbf{z}(t_i)$, the standard Kalman filter equations can be used to produce a perturbation estimate at time $[t))$, denoted as $\hat{\delta\mathbf{x}}(t)$. This value can be added to the nominal value calculated from the integration of $\mathbf{f}$ over the time period t_i, t_{i+1}, producing a total state estimate $\hat{\mathbf{x}}(t)$.

Immediately after the update step, the system is re-linearized about $\hat{\mathbf{x}}(t_i^+)$. A result of this re-linearization is that $\hat{\delta\mathbf{x}}(t/t_i)$ is $\mathbf{0}$ over the next time interval, meaning the best state estimate over this interval is reduced to:

$$\hat{\mathbf{x}}(t_{i+1}^-) = \hat{\mathbf{x}}(t_i^+) + \int_{t_i}^{t_{i+1}} \mathbf{f}[\hat{\mathbf{x}}(t/t_i), \mathbf{u}(t), t]dt \tag{2.56}$$

Equation 2.56 constitutes the discrete-time EKF state propagation equation. The covariance is propagated using

$$\mathbf{P}(t_{i+1}^-) = \mathbf{\Phi}[t_{i+1}, t_i; \hat{\mathbf{x}}(\tau/t_i)]\mathbf{P}(t_i^+)\mathbf{\Phi}^{\mathrm{T}}[t_{i+1}, t_i; \hat{\mathbf{x}}(\tau/t_i)]$$
$$+ \int_{t_i}^{t_{i+1}} \mathbf{\Phi}[t_{i+1}, i; \hat{\mathbf{x}}(\tau/t_i)]\mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^{\mathrm{T}}(t)\mathbf{\Phi}^{\mathrm{T}}[t_{i+1}, t; \hat{\mathbf{x}}(\tau/t_i)]dt \tag{2.57}$$

Finally, the EKF update equations are

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}[t_i; \hat{\mathbf{x}}(t_i^-)]\{\mathbf{H}[t_i; \hat{\mathbf{x}}(t_i^-)]\mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}[t_i; \hat{\mathbf{x}}(t_i^-)] + \mathbf{R}(t_i)\}^{-1} \tag{2.58}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\{\mathbf{z}_i - \mathbf{h}[\hat{\mathbf{x}}(t_i^-), t_i]\} \tag{2.59}$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}[t_i; \hat{\mathbf{x}}(t_i^-)]\mathbf{P}(t_i^-) \tag{2.60}$$

where $\mathbf{z}_i$ in equation 2.59 is the provided measurement, simulated or actual.

Two parameters that are useful in measuring Kalman filter performance are worthy of special mention. The first is the filter computed covariance, readily available as the result of Equation 2.60. The covariance $\mathbf{P}(t_i^+)$ takes the form of an $n \times n$ matrix, where n is the number of state variables. The off-diagonal terms represent the cross-covariance between state estimates, and the diagonal terms the variance associated with each individual state estimate. The square root of the variance, known as the standard deviation and represented

as $\boldsymbol{\sigma}$, gives a measure of the uncertainty associated with a given estimate. Specifically, given a Gaussian distribution and an estimated mean of μ, the true value of the estimated parameter lies within $\mu \pm \sigma$ with approximately 68% probability. The estimated mean is the *most likely* estimate of the parameter, however, the true value may fall anywhere within the distribution, and a higher variance results in a widening of the distribution with a corresponding lowering of the peak density value located at $x = \mu$. Essentially, a high variance means a less reliable estimate.

A problem lies in the fact that the filter computed covariance may not always mesh with reality. If the covariance of the state estimates are accurately computed by the estimator, the uncertainty may be accounted for in the application of the state estimates in other uses, such as for navigation. If, however, the filter under reports the true covariance values, the result is an inappropriate level of confidence in the accuracy of the estimate on the part of the user. It is therefore desirable to monitor the filter computed covariance in individual applications and compare it with true covariance, $\mathbf{P}_{r,true}$. The true covariance may be obtained through *covariance analysis* in the case of the standard Kalman filter; for the EKF, such analysis is not possible, and Monte Carlo methods must be applied to obtain $\mathbf{P}_{r,true}$. A description of covariance analysis and the reasons for which it may not be applied to the EKF are given in Section 2.6.

The second set of metrics that merits discussion are the *measurement residuals*, which are the difference between true measurement values and the filter's best estimate of what the measurements should be, before the measurement is actually taken [14]. The measurement residuals at time t_i are computed by subtracting from the measurements the a priori states after they have been passed through the measurement equation. The residuals

$$\mathbf{r}(t_i) = \mathbf{z}(t_i) - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-) \tag{2.61}$$

may be examined to determine the fitness of the sensors and system dynamics model [14], [15]. The a priori estimates $\hat{\mathbf{x}}(t_i^-)$ represent the state vector after propagation and

immediately preceding a measurement update. Frequent, large magnitude measurement residuals may correspond to a mismatch between actual measurements and the estimated best guess obtained through propagation. Alternatively, consistently large measurement residuals corresponding to a specific state may be an indication of sensor failure. The determination of what constitutes a *large* residual may be made by comparing each residual against the residual covariance

$$\mathbf{P}_r(t_i) = \mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) + \mathbf{R}(t_i) \tag{2.62}$$

As the residual sequence is white, Gaussian, and of zero mean [14], a comparison between the residual magnitude and a bound chosen based upon the residual covariance, such as 3σ, may be made to decide if a given measurement is erroneous.

2.6 Covariance Analysis

The characterization and analysis of a given Kalman filter is an important step that must be accomplished before such a filter is used in practical designs. In general, this will involve comparing the mean and covariance of estimated filter states against 'truth' values. In practice, actual truth will not be available, and reference data for testing purposes will be generated by the designer. This frequently involves many assumptions and simplifications of the system models, such as purely deterministic inputs, lack of wheel slippage, etc. The designer will then process real or simulated measurements through the filter and compare against the truth data. Doing this many times and computing ensemble statistics across realizations results in a Monte Carlo analysis. Although it is a widely used method, it can be time consuming, especially if actual data is desired and must thus be collected.

Another method that can be used to characterize a filter, without the need for measurements or the calculation of state estimates, is covariance analysis. This method is particularly useful when a picture of the effects of filter modification is desired. Reference

[14] provides a detailed description of covariance analysis using an error-state filter, and the following overview is based on that work.

Essentially, a filter is constructed that takes into account all required states, system dynamics and error sources. This model serves as the truth source, mimicking reality as closely as possible. Next, a suboptimal filter is created, based on a design model, often with fewer or combined states, neglected error sources, etc. These are then combined into augmented matrices, and propagated in time. The resulting augmented state covariance and estimation error covariance matrices can be observed for the level of divergence from the truth model.

The application of covariance analysis relies on the fact that the truth model is a linear system driven by white Gaussian noise, generating linear measurements corrupted by white Gaussian noise [14]. In the case of the EKF, the traditional method of covariance analysis cannot be applied. This is due to the fact that the EKF requires the calculation of state estimates for the linearization of the state transition matrix at each time step. Thus it is not possible to propagate the covariance matrix without the existence of measurements. It is therefore necessary to perform a Monte Carlo analysis for the characterization of the EKF. It the absence of actual measurements, the Monte Carlo analysis is based on simulated data, which may not always result in an accurate portrayal of filter response. The focus of this work is to characterize EKF-based image-aided navigation using Monte Carlo methods and to provide a baseline against which individual filter realizations may be compared.

2.7 Histograms and PDF Fitting

2.7.1 Histogram. The foundation of this research is the analysis of filter committed errors in position and heading as compared to truth, obtained by navigating a ground vehicle over a level surface. An incomplete, but easy to calculate and interpret description of filter errors may be had by deriving a histogram of the errors. The histogram

is essentially a bar graph of groupings of data, where the x-axis describes the magnitude of the data range, and the y-axis the number of data points that fall within a selected range. Such a representation allows for a rough estimation of the probability distribution of a data set, but does result in some loss of information [17]. The effectiveness of the method is dependent on both the width of the bins in which the data is sorted and the chosen endpoints of the bins. Histograms produced in this work will cover the range between the minimum and maximum data points, and the space between divided into a number of bins equal to the number of data points being analyzed. Though it is possible to attempt to fit a density function directly to a histogram, the loss of information incurred by binning the data will lead to a substandard result. As such, histograms are used only as a visual aid in this work. If Gaussian error distributions are assumed, and PDF fitting to the error data performed, comparison of the resulting density functions with their corresponding histograms can help to determine the appropriateness of the Gaussian fit.

2.7.2 PDF Fitting. A more complete description of error distribution than that provided by the histogram may be given by specifying the probability density function (PDF) of the errors at each point in time. In the case of a Gaussian distribution, the PDF of the states is completely described by the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$, both values provided in the Kalman filter equations. With respect to the description of *errors*, the true states are known with certainty, and may be viewed as Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\sigma}^2 = \mathbf{0}$. As the EKF maintains Gaussian distributions of the estimates, the linear combination

$$\mathbf{e}(t_i) = \hat{\mathbf{x}}(t/t_i) - \mathbf{x}(t_i) \tag{2.63}$$

is also Gaussian distributed and completely specified by the corresponding mean and covariance. Thus, in determining the PDF of the errors committed by the filter, a Gaussian

distribution is assumed, and the mean

$$\boldsymbol{\mu}_e(t_i) = \mathrm{E}[\mathbf{e}(t_i)] = \frac{1}{n}\sum_{k=1}^{n}[\hat{\mathbf{x}}_k(t/t_i) - \mathbf{x}_k(t_i)] \tag{2.64}$$

and covariance

$$\mathbf{P}_e(t_i) = \mathrm{E}[(\mathbf{e}(t_i) - \boldsymbol{\mu}_e(t_i))(\mathbf{e}(t_i) - \boldsymbol{\mu}_e(t_i))^{\mathrm{T}}] \tag{2.65}$$

at each timestep are calculated using Equations 2.64 and 2.65, respectively.

2.7.3 Entropy. Upon the determination of the true PDF of the errors committed by the filter using the methods described in Section 2.7.2, it becomes necessary to determine a measure of accuracy of the error model as compared with the truth. A popular method for doing so is the Normalized Estimation Error Squared (NEES), defined as

$$\eta(t_i) = [\hat{\mathbf{x}}(t/t_i) - \mathbf{x}(t_i)]^{\mathrm{T}}\mathbf{P}(t/t_i)^{-1}[\hat{\mathbf{x}}(t/t_i) - \mathbf{x}(t_i)] \tag{2.66}$$

However, application of the NEES test of filter consistency requires that the system under consideration is linear and subject to zero-mean white Gaussian process and measurement noise [2]. While the NEES measurement in general may be used to evaluate the consistency of the EKF-predicted distributions, it is not capable of handling non-Gaussian distributions, should they be encountered. In these cases, the *discrete entropy* measure may be used, as suggested by Taylor [20]. Entropy is a measure of the uncertainty in a system whose states are described by a given probability distribution [18].

The total entropy of a distribution is

$$E = -\sum_{\mathrm{i}=1}^{\mathrm{k}} p_i \log_{\mathrm{b}} p_i \tag{2.67}$$

Equation 2.67 is also known as the Boltzmann-Gibbs-Shannon measure of entropy [18]. Given the location of a true data point, a selected parcel of the surrounding area is divided into k blocks. The probability p_i is the probability of the true data point residing in block i of the distribution being tested. The base of the logarithm b may be arbitrarily chosen, so long as it is kept consistent between tests.

2.8 Camera Modeling and Calibration

In image-aided navigation, imaging sensors are used to capture pictures of the area in the vicinity of the vehicle. From these images, features are extracted, and the change in the location of the features from frame to frame is used to derive the relative motion of the vehicle. The images produced are a two-dimensional representation of a three-dimensional world, and this is a process of projection during which one dimension is lost [5]. However, the mapping is not generally an accurate two-dimensional representation of reality; distortions of the image can be introduced through lens imperfections, misalignment, or other structural or manufacturing issues. The quantification of the effects of these distortions upon an image is found in a collection of descriptors known as intrinsic parameters. In addition, to derive relative movement from features in the images, camera translations and rotations with respect to the scene must be accounted for. This is done through the modeling of extrinsic parameters. The aim of camera calibration is to obtain mathematical descriptors for the intrinsic and extrinsic parameters. Combining these with the three-dimensional to two-dimensional mapping function results in a complete camera model. Section 2.8.1 will address the mapping of three-dimensional images to the two-dimensional image plane, and Section 2.8.2 will present the models for lens distortions. The results of Sections 2.8.1 and 2.8.2 will be combined to produce a complete distortion model containing the intrinsic parameters, which is presented in 2.8.3. Camera rotations and translations, described through the extrinsic parameters, will be discussed in Section 2.8.4. Finally, Section 2.8.5 will cover the process of camera calibration using the Caltech Camera Calibration Toolbox (CCCT) for MATLAB®, developed by Jean Bouguet [3].

2.8.1 Mapping. In the image-processing that was completed during the course of this work, the pinhole camera model was employed to described the way in which light is collected by the camera and the image projected. This is simply a foundational model, and

distortion effects will be accounted for separately. In this model, the aperture of the camera (also known as the optical center) is modeled as a single point through which all light rays pass, and camera focus and lens thickness are ignored [5]. The true image plane lies at a distance f from the optical center along the $-z_c$ axis and is parallel with the xy plane of the c coordinate frame. A point S in $\mathbb{R}^3$ is projected onto the true image plane where the line passing through the origin of the c-frame to point S intersects the true image plane. The projection onto the true image plane of all points that represent a scene will appear inverted when compared to the original scene. To rectify this, a construct known as the virtual image plane is employed, parallel to the true image plane, but intersecting the z_c axis at the distance f from the optical center. The projection of the point S onto the virtual image frame is denoted as $\mathbf{s}^{\mathbf{c}}_{\mathbf{proj}}$, and is located at the point where the vector describing the location of the point S in the camera frame, $\mathbf{s}^c$, intersects the virtual image plane. Such a projection is not inverted when compared to the original scene. The concept of the virtual image plane is illustrated in Figure 2.4. By the geometry of similar triangles [5], it can be seen that the mapping of image points in $\mathbb{R}^3$ Euclidean space, expressed in c-frame coordinates, to $\mathbb{R}^2$ Euclidean space is

$$[\mathbf{v}^{\mathbf{c}}_x, \mathbf{v}^{\mathbf{c}}_y, \mathbf{v}^{\mathbf{c}}_z]^{\mathrm{T}} \overset{\mathbb{R}^3:\mathbb{R}^2}{\rightarrow} [f\mathbf{v}^{\mathbf{c}}_x/\mathbf{v}^{\mathbf{c}}_z, f\mathbf{v}^{\mathbf{c}}_y/\mathbf{v}^{\mathbf{c}}_z]^{\mathrm{T}} \tag{2.68}$$

or in a vector notation as in [9], where it is shown as

$$\mathbf{s}^c_{proj} = \frac{f}{\mathbf{s}^c_z}\mathbf{s}^c \tag{2.69}$$

Application of Equation 2.69 results in the vector $\mathbf{s}^c$, which describes the location of the point S in the c-frame, being expressed in a new, transitionary coordinate frame, denoted as the projected c-frame, shown in Figure 2.5. However, the required expression of this point in *pix*-frame coordinates remains to be found. There are four separate aspects to this coordinate transformation. First, as the z coordinate of all points on the image plane is equal to the focal length f, only x and y coordinates are required to describe a projected

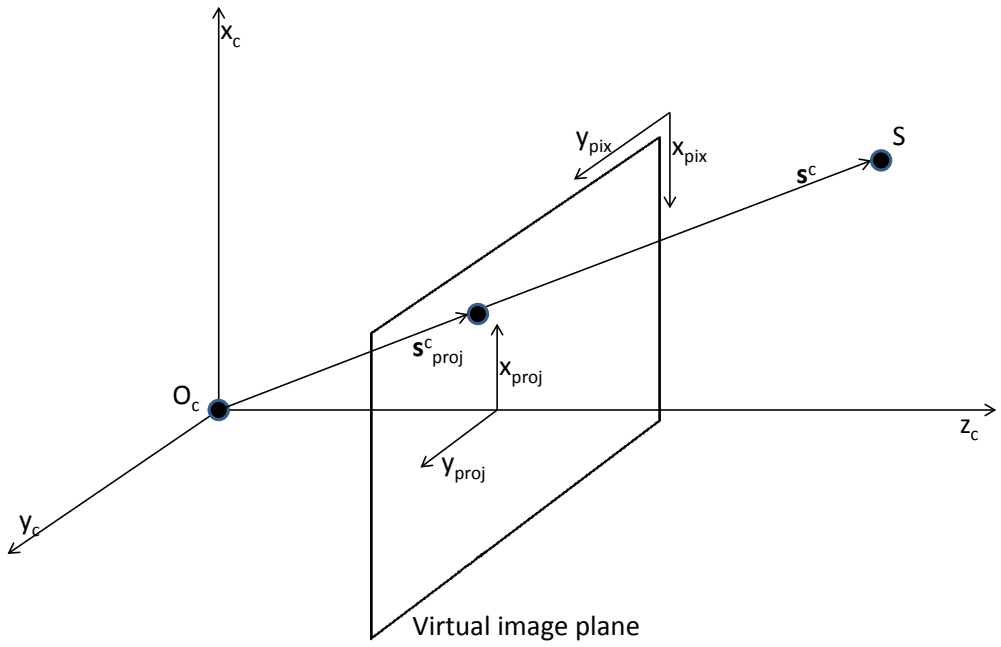

Figure 2.4: The pinhole camera model. The camera aperture captures light reflected from an object in three-dimensional space. The image of the object is projected on the virtual image plane at the point where the vector $\mathbf{s}^c$ from the camera aperture $\mathbf{O}_c$ to the object S intersects the virtual image plane.

image point $\mathbf{s}^{proj}$ in the *pix*-frame. Further, the *x*-axis in the *c*-frame is defined in the opposite direction as the *x*-axis in the *pix*-frame. There is also a scaling operation that must be considered as the image is projected onto the fixed-sized *pix*-frame. Finally, there is a coordinate shift, due to the origin of the projected *c*-frame axes being defined as having a different location from that of the *pix*-frame. The derivation of this process follows, from [9].

As stated in the above paragraph, the *z*-coordinate of the $\mathbf{s}^c_{proj}$ is not required in the description of the point in either the two-dimensional projected *c*-frame or the *pix*-frame. Thus, after the scaling operation shown in Equation 2.69, a projected image point $\mathbf{s}^c_{proj}$ can be pre-multiplied by a matrix to remove the *z* coordinate

$$\mathbf{s}^{proj} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{s}^c_{proj} \tag{2.70}$$

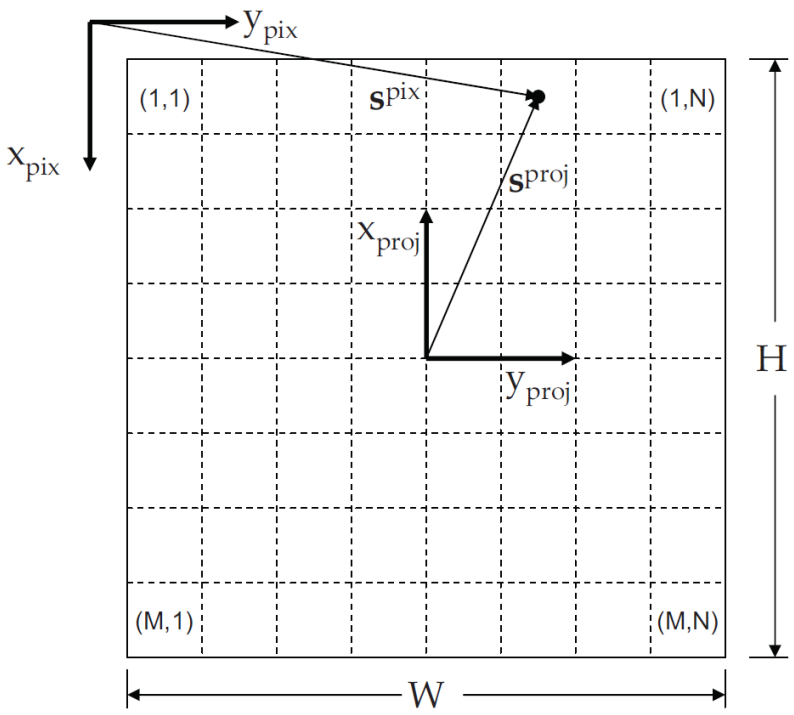

Figure 2.5: The image plane has a physical width W and height H, as well as pixel width N and height M.The *pix*-frame axes are shown in the upper left corner, and the projected c-frame axes are shown with origin at $(\frac{M+1}{2}, \frac{N+1}{2})$ [9][16].

The result of this operation, $\mathbf{s}^{proj}$, is two-dimensional projection of $\mathbf{s}^{c}_{proj}$ onto the *pix*-frame. The removal of the z-coordinate being completed, it is then necessary to change the sign on the x-coordinate of the projected point as it is converted to a description in *pix*-frame coordinates. As this only entails multiplication of the x-coordinate by a negative, it can be represented by another matrix pre-multiplication

$$\mathbf{s}^{pix'} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{s}^{proj} \tag{2.71}$$

The term $\mathbf{s}^{proj}$ in Equation 2.71 refers to a point $\mathbf{p}$ in the projected c-frame, while $\mathbf{s}^{pix}$ is the same point in the transitionary *pix'*-frame. The notation *pix'* simply denotes that the

point is not yet fully described in *pix*-frame coordinates, having not yet undergone the scaling and translation operations.

The scaling of the image to the *pix*-frame depends on two factors: the physical dimensions of the pixel array, and the pixel arrangement. Pixel arrangement refers to the number of pixels in the rows and columns of the pixel array, and assumes a rectangular array. The shifting of the image coordinates relies only on the arrangement. Denoting H as the physical height of the array, W as the width, M as the number of pixels along the x-axis of the *pix*-frame, and N as the number of pixels along the y-axis, it can be seen that the distance between the c-frame origin and the *pix*-frame origin is $\frac{M+1}{2}$ in the x-direction and $\frac{N+1}{2}$ in the y-direction. Therefore, the following expression for the scaling and shifting operation can be described:

$$\mathbf{s}^{pix} = \begin{bmatrix} \frac{M}{H} & 0 \\ 0 & \frac{N}{W} \end{bmatrix} \mathbf{s}^{c}_{proj} + \begin{bmatrix} \frac{M+1}{2} \\ \frac{N+1}{2} \end{bmatrix} \tag{2.72}$$

The results of Equations 2.69, 2.70, 2.71 and 2.72 can be combined into a transformation matrix that enables the transformation of a vector of coordinates in the c-frame, $\mathbf{s}^c$, into a homogeneous vector of coordinates in the *pix*-frame

$$\underline{\mathbf{s}}^{pix} = \frac{1}{\mathbf{s}^c_z} \mathbf{T}^{pix}_c \mathbf{s}^c \tag{2.73}$$

where

$$\mathbf{T}^{pix}_c = \begin{bmatrix} -f\frac{M}{H} & 0 & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.74}$$

The inversion of Equation 2.74, $\mathbf{T}^c_{pix}$, can be used to transform *pix*-frame coordinates into the c-frame:

$$\underline{\mathbf{s}}^c = \mathbf{T}^c_{pix} \mathbf{s}^{pix} \tag{2.75}$$

where

$$\mathbf{T}^{c}_{pix} = \begin{bmatrix} -\frac{H}{fM} & 0 & \frac{H(M+1)}{2fM} \\ 0 & \frac{W}{fN} & -\frac{W(N+1)}{2fN} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.76}$$

However, the z-coordinate $\mathbf{s}^{c}_{z}$ cannot be restored through this method, as the required information was removed. However, principles of epipolar geometry can be applied to sets of images taken from two cameras at the same instant in time to recover this lost coordinate [19]. Therefore, the result of Equation 2.75, $\underline{\mathbf{s}}^{c}$, is a homogeneous vector that is co-directional with, but may not be equal to, vector $\mathbf{s}^{c}$ [9].

2.8.2 Image Distortion Models. The intrinsic parameters of an imaging sensor include the focal length, the principal point, and the lens distortion parameters. The focal length f, mentioned in Section 2.8.1, is the distance between the optical center (the origin of the c-frame) and the image plane. The principal point $\mathbf{p}$ is ideally placed at the geometric center of the image plane (Section 2.8.2.2 addresses when this may be untrue). The lens distortion parameters presented here as in [3], include radial distortion, tangential distortion, and skew.

2.8.2.1 Radial Distortion. Radial distortion is the most prominent distortion effect [26], which is manifested in the curving of lines within an image. Radial distortion is caused by the bending of optical rays as they pass through camera lenses [10]. The expression for radial distortion is a non-linear approximation, dependent on the distance between the origin of the c-frame and coordinates of projected point $\mathbf{s}^{proj}$. This distance is given by

$$r = \sqrt{(\mathbf{s}^{proj}_{x})^2 + (\mathbf{s}^{proj}_{y})^2} \tag{2.77}$$

and the radial distortion is expressed as

$$d^{rad} = 1 + k_{\mathrm{r}1} r^2 + k_{\mathrm{r}2} r^4 + k_{\mathrm{r}3} r^6 \tag{2.78}$$

The terms k_{r1}, k_{r2} and k_{r3} are constant distortion coefficients. Zhang [26] finds the radial distortion to be appropriately modeled using only the first two coefficients; however, three terms are applied within the CCCT software that is used in this research, and thus this is the form presented.

2.8.2.2 Tangential Distortion. Tangential distortion is typically caused when the centers of curvature of lens surfaces are not collinear [6]. This has the effect of decentering the principal point from the center of the image plane [9], and is a function of tangential distortion constants k_{t1} and k_{t2}, as well as the distance r and projected coordinates $\mathbf{s}^{proj}$. The expression for the tangential distortion vector is

$$\begin{bmatrix} d_x^{tan} \\ d_y^{tan} \end{bmatrix} = \begin{bmatrix} 2k_{t1}(\mathbf{s}_x^{proj})(\mathbf{s}_y^{proj}) + k_{t2}[r^2 + 2(\mathbf{s}_x^{proj})^2] \\ k_{t1}[r^2 + 2(\mathbf{s}_y^{proj})^2 + 2k_{t2}(\mathbf{s}_x^{proj})(\mathbf{s}_y^{proj})] \end{bmatrix} \tag{2.79}$$

2.8.2.3 Skew. The skew coefficient a_c is a scalar measure of how much the angle between the x and y axes of the *pix*-frame differs from $\frac{\pi}{2}$ radians. For perfectly orthogonal axes, meaning rectangular pixels, a_c assumes a value of 0. The angle between the x and y axes of the *pix*-frame is

$$\Theta_{xy}^{pix} = \frac{\pi}{2} - \arctan(a_c) \tag{2.80}$$

In modern imaging sensors, the amount of skew in an image tends to be very close to zero, and thus it is often neglected [3]. The skew distortion was not taken into account for this work, but is included in the final distortion models for completeness. In the sense of pixel coordinates, the skew coefficient provides for an adjustment of the x coordinate of $\underline{\mathbf{s}}^{pix}$ by modifying the upper row of the matrix $\mathbf{T}_c^{pix}$, resulting in the camera intrinsic matrix

$$\mathbf{T}_c^{pix} = \begin{bmatrix} -f\frac{M}{H} & -a_c f\frac{M}{H} & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.81}$$

2.8.3 Complete Camera Model. Having presented expressions for both the projection of a point $\mathbf{s}^c$ onto the idealized *pix*-frame and the distortions introduced by the camera itself, a complete representation of the mapping of a point vector within the *c*-frame to a homogeneous vector in the *pix*-frame can be given as

$$\underline{\mathbf{s}}^{pix} = \frac{1}{\mathbf{s}_z^c}\begin{bmatrix} -f\frac{M}{H} & -a_c f\frac{M}{H} & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} d^{rad} & 0 & d_x^{tan} \\ 0 & d^{rad} & d_y^{tan} \\ 0 & 0 & 1 \end{bmatrix}\mathbf{s}^c \tag{2.82}$$

2.8.4 Camera Translations and Rotations. Neglecting radial and tangential distortions, the mapping of a three-dimensional world coordinate to the two dimensional *pix*-frame is given by

$$\alpha\underline{\mathbf{s}}^{pix} = \mathbf{T}_c^{pix}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\underline{\mathbf{m}} \tag{2.83}$$

where α is an arbitrary scale factor, $\underline{\mathbf{m}}$ is a homogeneous vector describing a point in the world coordinate system, and $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ is an augmented matrix containing the extrinsic parameters. $\mathbf{T}_c^{pix}$ is as presented in Equation 2.81.

The extrinsic parameters relate the world coordinate system to the camera coordinate system through a rotation and translation [26]. These values are stored within the augmented matrix

$$\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.84}$$

Denoting each of the first three columns of the matrix in Equation 2.84 by $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$ respectively, $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ can be written as $\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}$.

All homogeneous three-dimensional coordinates $\underline{\mathbf{m}}$ are assumed to lie on the $z = 0$ plane of the world coordinate frame. Doing so does not in any way reduce generality [26], and the $\underline{\mathbf{m}}_z$ term may therefore be dropped. This also allows for the reduction of the

matrix in Equation 2.84 through the removal of the third row and column. Therefore, the homogeneous vector $\underline{\mathbf{m}}$ and the extrinsic matrix $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ are redefined as

$$\underline{\mathbf{m}} = \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} \tag{2.85}$$

and

$$\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix} \tag{2.86}$$

A result of Equation 2.85 is that the i^{th} column of the rotational portion of the matrix $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ becomes

$$\mathbf{r}_i = \begin{bmatrix} r_{1i} \\ r_{2i} \\ 0 \end{bmatrix} \tag{2.87}$$

Given these reductions, a *homography* **H**, which is a function that maps lines to lines while operating on homogeneous two-dimensional vectors [5], can be defined as

$$\mathbf{H} = \mathbf{T}_{\mathbf{c}}^{pix} \begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix} \tag{2.88}$$

Comparing Equations 2.88 and 2.83, the relationship between $\underline{\mathbf{m}}$ and $\underline{\mathbf{s}}^{pix}$ is

$$\alpha \underline{\mathbf{s}}^{pix} = \mathbf{H}\underline{\mathbf{m}} \tag{2.89}$$

2.8.5 Camera Calibration. Having described the intrinsic and extrinsic parameters, camera calibration is now described. Camera calibration is the process of solving for the general homography **H** as shown in Equation 2.88. The use of the CCCT requires a square-checked planar calibration board, the number and dimensions of the squares being known, as the calibration surface. Images of this surface are captured by the cameras, with a translation and rotation of either the calibration surface or the camera rig occurring between each image capture. At least two sets of images are required. The

image sets are passed to the software, and the outer corners of the calibration surface are manually identified by the user.

The algorithm used to solve for the intrinsic and extrinsic parameters is based on methods described in [26], adopted for the intrinsic matrix presented in [6] and discussed in Section 2.8.2. The process begins with by solving for the maximum likelihood estimation of homography **H**, which is used to constrain the intrinsic parameters using methods described in [26]. A closed-form estimated solution for **H** is computed using the orthogonality of vanishing points, and includes the distortion terms defining radial and tangential distortions [3]. The final step, also from Zhang [26], is refinement of the estimate of **H** by minimizing the functional

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \| \mathbf{s}_{ij}^{pix} - \hat{\mathbf{s}}^{pix}(\mathbf{T}_{c}^{pix}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{m}_j) \|^2 \tag{2.90}$$

Minimization is accomplished through application of of the Levenberg-Marquardt algorithm, a description of which can be found in [12].

2.9 Image-Aided Navigation

This section provides an overview of the deeply-coupled image-aided inertial navigation algorithm implemented in this research, developed by Veth [25]. First, a description of Lowe's Scale-Invariant Feature Transform (SIFT) [11] is given. Application of this method results in descriptors of image features, which are matched between images collected by separate cameras over time. This information can be leveraged to determine the relative change in position of the platform between image updates. Next, feature matching algorithms and the determination of the depth coordinate of a real-world point from a pair of simultaneously captured two-dimensional images are discussed. Then, a presentation of the stochastic feature tracker follows, which allows for the prediction of feature location regions in images. The final portion presents the process by which inertial and image data are coupled to provide a navigation solution.

2.9.1 SIFT Algorithm. The SIFT algorithm [11] enables the extraction of feature keypoints from images, and provides for a description of individual points in a manner that is invariant to changes in image scale and rotation, as well as partially invariant to changes in illumination and affine distortion resulting from differing camera viewpoints. The algorithm is divided into 4 steps: scale-space extrema detection, keypoint localization, orientation assignment, and computation of the keypoint descriptor. Each of these steps are discussed in detail below.

2.9.1.1 Scale-Space Extrema Detection. The detection of potential keypoints within an image begins with locating stable features, those that can be reliably extracted across all possible scales. The method proposed by Lowe for determining these features involves first smoothing the images through convolution with a variable-scale Gaussian function

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.91}$$

where the Gaussian function is given by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-(x^2+y^2)/\sigma^2} \tag{2.92}$$

Multiple smoothed images of differing scales are produced using a varied σ, obtained by multiplying σ by a scale factor k. An image can be subtracted from the adjacent image with a higher k value to produce the difference-of-Gaussian (DG) for the two images. This operation is expressed as

$$D(x, y, \sigma) = (G(x, y, k\sigma)) - (G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.93}$$

This process is depicted visually in Figure 2.6. The DG function is chosen as it is a close approximation to the Laplacian of Gaussian, the extrema of which have been shown to result in the most stable image features as compared to a number of other proposed functions [11]. Once the DGs are computed for the scale range from σ to 2σ, or octave,

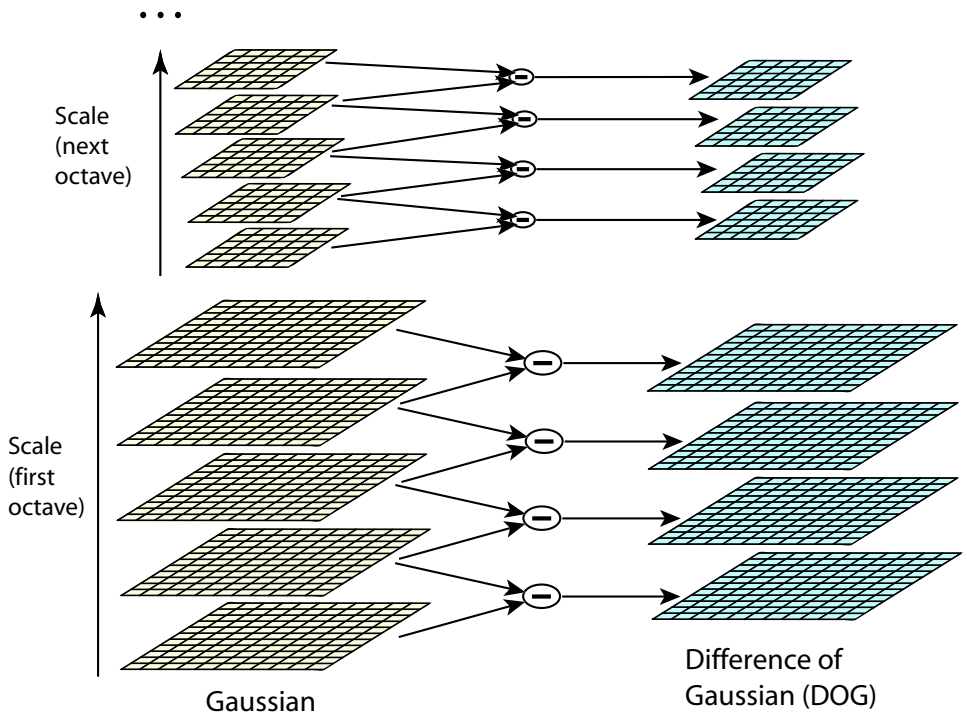

Figure 2.6: The Difference-of-Gaussian function. The left column represents 2 octaves of Gaussian smoothed images. Subtraction of adjacent images results in the difference-of-Gaussian, depicted in the right column [11].

the image convolved with the Gaussian of 2σ is downsampled by a factor of two, and the process repeated. Local maxima and minima are determined by comparing a sample point with the 8 nearest points within the same scale, and the 9 adjacent points in the next highest and lowest scales, as illustrated in Figure 2.7. The sample point must be greater or lesser than all of these neighboring points to be considered a candidate feature.

2.9.1.2 Keypoint Localization. Not all candidates found through extrema detection are well-suited for use in feature matching. Points that are sensitive to noise due to low contrast or poorly localized should be removed from further consideration. Points having such undesirable qualities are detected using two separate methods: 3D interpolation, and analysis of the principal curvatures.

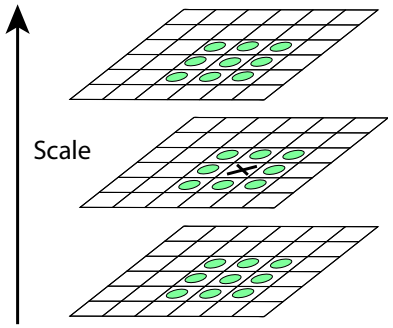

Figure 2.7: Extrema detection. A sample point is compared with the 8 surrounding points in the sample scale, as well as the 9 adjacent points in the scales directly above and below the sample scale [11].

Rather than locate keypoints at the same scale and location as the sample point from which they are derived, improved matching and stability can be obtained through the use of 3D interpolation [11]. Using the second-order Taylor series expansion of Equation 2.93, evaluated at the sample point and shifted by $\mathbf{x} = [x, y, \sigma]^{\mathrm{T}}$ so that the origin is at the sample point

$$D(\mathbf{x}) = D + \frac{\delta D^{\mathrm{T}}}{\delta \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\frac{\delta^2 D}{\delta \mathbf{x}^2}\mathbf{x} \tag{2.94}$$

Taking the derivative of Equation 2.94 and setting equal to $\mathbf{0}$ results in an expression for the extremum location offset $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = -\frac{\delta^2 D^{-1}}{\delta \mathbf{x}^2}\frac{\delta D}{\delta \mathbf{x}} \tag{2.95}$$

An offset found through Equation 2.95 that is greater than 0.5 in any dimension indicates that the extremum is actually closer to a different point. Interpolation would then be conducted about this other point, and $\hat{\mathbf{x}}$ added to this sample point to give the final location of the extremum. Equation 2.94 can be modified by incorporating Equation 2.95 through substitution, resulting in the DG function value at the extrema location:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\delta D^{\mathrm{T}}}{\delta \mathbf{x}}\hat{\mathbf{x}} \tag{2.96}$$

The result of Equation 2.96 can be compared to a pre-determined contrast threshold value, and the point rejected if this criteria is not met.

To further improve stability, it is also necessary to remove points that are located on edges and poorly determined. A poorly defined peak in the DG function will have a large principal curvature across the edge but a small one in the perpendicular direction [11]. The principal curvatures are calculated by evaluating the 2×2 Hessian matrix $\mathbf{H}$

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.97}$$

where the derivatives are estimated by finding the differences between the keypoint and adjacent sample points. The ratio of the eigenvalues of $\mathbf{H}$ is proportional to the principal curvatures. Denoting r as the threshold ratio between the larger and smaller eigenvalues, the following relationship is evaluated to compare the ratio of principal curvatures with r:

$$\frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - (D_{xy})^2} < \frac{(r+1)^2}{r} \tag{2.98}$$

A point is dismissed as a keypoint candidate if it does not satisfy Equation 2.98.

2.9.1.3 Orientation Assignment. To achieve descriptor invariance to image rotation, the keypoints which meet the contrast and ratio of principle curvature thresholds are assigned an orientation relative to the local image properties. The keypoint can then be described relative to this rotation, imparting the desired quality of invariance to rotation.

The determination of the local orientation vector begins with the calculation of the gradient magnitudes and orientations of sample points surrounding a given keypoint. The sample points are drawn from the smoothed image L that is closest in scale to the keypoint in question, and the magnitude and orientation angles are respectively calculated using the

following two equations:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{2.99}$$

$$\theta(x, y) = \arctan((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y))) \tag{2.100}$$

A histogram is constructed from the gradient orientations determined using Equation 2.100, each being weighted by its corresponding magnitude and a Gaussian circular window with a σ value 1.5 times that of the keypoint scale. An initial keypoint is defined from the highest peak in the histogram, and additional keypoints are created from any peaks within 80% of the peak value. It is thus possible to define multiple keypoints from one sample location, each identical in scale and location and differing in orientation. As a final step to improve accuracy, each peak location is adjusted by applying a parabolic interpolation over the three closest values to each peak.

2.9.1.4 Computation of the Keypoint Descriptor. Having determined keypoints locations, scales and orientations, the final step is to construct a descriptor to be assigned to each keypoint. This is accomplished by first sampling the image gradients around a keypoint, and rotating the coordinates of both the descriptor and gradient orientations relative to the keypoint orientation, accomplishing rotational invariance. A Gaussian weighting function is applied over the sampling area to reduce the contribution of gradients further from the keypoint. Next, orientation histograms are constructed from the weighted gradient orientations, each histogram containing 8 directional bins. Interpolation is then applied to distribute gradients into adjacent histogram bins to remove sudden changes in the descriptor due to movement of the sample point. The descriptor is formed from the values contained in the histogram bins. Lowe concludes through experimentation that the best results are obtained using a 4×4 array of histograms with 8 bins each, resulting in a 128 element descriptor for each keypoint [11]. A visual depiction of the determination of the keypoint descriptor is shown in Figure 2.8.

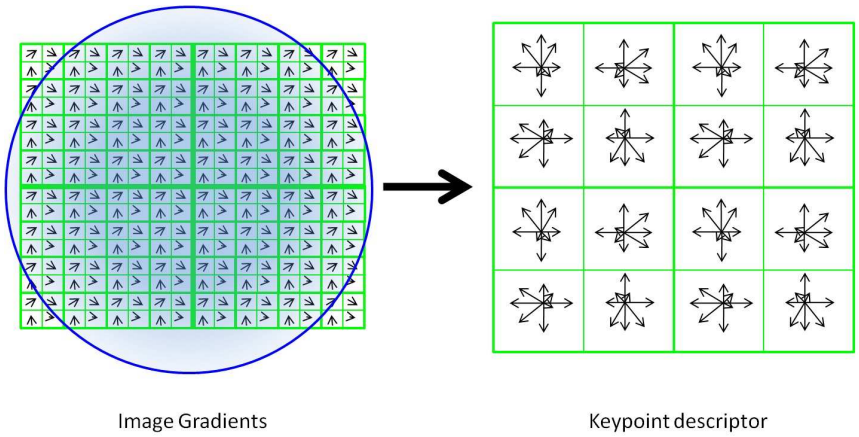

Figure 2.8: A visual depiction of the keypoint descriptor. A Gaussian weighting function is applied to the image gradients surrounding the keypoint location, which are then combined into histograms with 8 directional bins each. Interpolation over histograms spreads the influence of gradients into adjacent bins,and the descriptor is formed from the 128 gradient vectors contained within the histograms [11].

2.9.2 Epipolar Geometry. Once scale and rotation invariant features are selected within an image for use as keypoints, the next requirement is to establish a correspondence between points within two images taken at the same instance in time. The brute force method would be to simply compare (without constraint) all points within the first image to the keypoints extracted from the second and select each respective best match. However, the efficiency of this search can be improved by exploiting the knowledge of camera locations and orientations in space in combination with the feature location in the first image plane [5], [19]. This is an example application of epipolar geometry, and the method by which it is applied to feature matching follows. A pictorial depiction is given in Figure 2.9.

Assuming the relative locations and rotations of the cameras are known, a baseline ℓ_b can be constructed which connects the origins of the respective c-frames. Further, assume a point S in space has been captured in an image by camera 1, and thus is projected upon the virtual image plane of camera 1 at a location given by vector $\mathbf{s}^c_{proj1}$. It is easy to see that

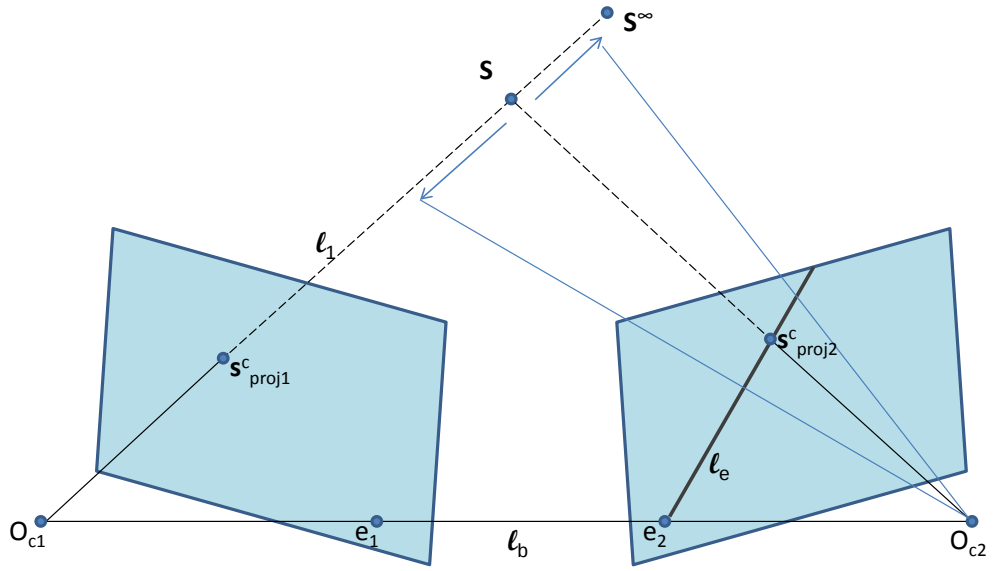

Figure 2.9: A depiction of the epipolar geometry concept. Knowledge of the origins of each camera frame can be used to restrict the search for feature correspondence between two images to a line in the second image. This line is known as the epipolar line, and is denoted by ℓ_e.

no depth information is conveyed by $\mathbf{s}^c_{proj1}$; the three-dimensional location of point S can only be restricted to a line vector $\ell_\mathbf{1}$ that extends from the origin of camera 1 through $\mathbf{s}^c_{proj1}$, and terminating at infinity. However, the lines ℓ_b and $\ell_\mathbf{1}$ describe an epipolar plane which intersects both virtual image planes. The projection of point **S** onto the image plane of camera 2 is required to lie upon the line which describes the intersection of the epipolar plane with the virtual image plane of camera 2. Thus in feature matching the search for a corresponding point can be restricted to this line, resulting in a much improved search algorithm in terms of efficiency.

2.9.3 Predicted Feature Matching. The IAN algorithm considered in this work makes use of the stochastic feature tracker developed by Veth [25] to predict the location of features within the next set of collected images. In general, the predicted pixel location and covariance of a feature at time t_{i+1} is a non-linear function of the vehicle position in

the navigation frame $\mathbf{p}^n$, the body-to-navigation frame DCM $\mathbf{C}_b^n$, the landmark locations in the navigation frame $\mathbf{y}^n$, the camera-to-body DCM $\mathbf{C}_c^b$, and the camera-to-pixel frame transformation matrix $\mathbf{T}_c^{pix}$

$$\mathbf{z}(t_{i+1}) = \mathbf{h}(\mathbf{p}^n(t_{i+1}), \mathbf{C}_b^n(t_{i+1}), \mathbf{y}^n(t_{i+1}), \mathbf{C}_c^b, \mathbf{T}_c^{pix}) \tag{2.101}$$

The derivation of the statistical prediction equations will not be reproduced here; the details may be found in [24]. The results of these equations are used to assign the estimated pixel locations and covariances of features in an image taken at a later time. The covariance ellipsoid associated with this estimate is used to constrain the area of search for a match. Once a new set of images is introduced, each SIFT feature found within the new image is compared with all predicted feature locations by computing the Mahanabolis distance between the measured location and the predicted locations. For an image captured at time t_i, the Mahanabolis distance between a measured location $\mathbf{d}(t_i)$ and a predicted location $\hat{\mathbf{d}}(t_i)$ is defined as

$$D(t_i) = [\mathbf{d}(t_i) - \hat{\mathbf{d}}(t_i)]^{\mathrm{T}} \mathbf{P}_{zz}(t_i) [\mathbf{d}(t_i) - \hat{\mathbf{d}}(t_i)] \tag{2.102}$$

All distances that fall below a predetermined constraint correspond to candidate matches. The feature descriptor vector for each of these candidates is then compared to the descriptor of the measured point through the computation of the dot product between the two. A match is declared if the largest magnitude dot product for all predicted points is greater than a user identified minimum correlation measure and is sufficiently distinct from the predicted point with the next highest correlation.

2.9.4 Depth Coordinate Determination. The depth of the three-dimensional point S can be calculated after establishment of a relationship between the focal lengths of the cameras and the disparities between pixel locations. As described by Szeliski [19], this can be accomplished by rectifying the images in question. The first step in rectification is to apply a rotation to each camera frame so that the respective z_c-axes of each camera

frame are perpendicular to line ℓ_b connecting the camera origins. Another rotation about ℓ_b is performed so that the cameras' x_c-axes are perpendicular to ℓ_b. Finally, the smaller image is magnified to account for differing focal lengths. The depth coordinate Z_c of point S can then be determined using

$$d = f\frac{\ell}{Z_c} \tag{2.103}$$

where f is the focal length, ℓ is the length of the baseline ℓ_b, and d is the disparity between respective feature coordinates for each camera

$$d(x, y) = y_1 - y_2;\ x_1 = x_2 \tag{2.104}$$

2.10 Literature Review

The purpose of this work is to characterize the errors in IAN. As preparation for this endeavor, a literature review was conducted, focusing on publications which consider the specific algorithm used, and error sources common to algorithms which rely on building a stochastic map of landmarks to perform localization.

2.10.1 Veth. This section provides an overview of the deeply coupled image-aided inertial navigation algorithm [23]. A visual depiction of the system is shown in Figure 2.10. This system is based upon an error state EKF which integrates information from a micro-electro-mechanical system (MEMS) grade IMU and a pair of cameras rigidly mounted to a frame to compute a navigation solution. The filter tracks the errors in position $\delta\mathbf{p}^n$ and velocity $\delta\mathbf{v}^n$ vectors, angles about the North, East and Down axes $\delta\boldsymbol{\psi}$, the biases on the accelerometers, $\delta\mathbf{a}^{b_0}$ and gyroscopes, $\delta\mathbf{b}^{b_0}$, and the m tracked feature locations $\mathbf{t}^e_{\mathrm{m}}$ [25]. Collecting all terms, the error state vector is depicted as

$$\delta\hat{\mathbf{x}} = \left[\delta\mathbf{p}^n\ \delta\mathbf{v}^n\ \delta\boldsymbol{\psi}\ \delta\mathbf{a}^{b_0}\ \delta\mathbf{b}^{b_0}\ \mathbf{t}^e_m\right]^{\mathrm{T}} \tag{2.105}$$

Within the EKF, $\delta\mathbf{p}^n$, $\delta\mathbf{v}^n$, and $\delta\boldsymbol{\psi}$ are all modeled as stochastic processes, the biases $\delta\mathbf{a}^{b_0}$ and $\delta\mathbf{b}^{b_0}$ are modeled as first-order Gauss-Markov processes, and all landmarks are considered stationary.

In the error state EKF, the final state estimates $\overline{\mathbf{x}}(t_i^+)$ are calculated as the sum of the post update state error vector shown in Equation 2.105 and a nominal trajectory $\overline{\mathbf{x}}(t_i^-)$

$$\overline{\mathbf{x}}(t_i^+) = \overline{\mathbf{x}}(t_i^-) + \delta\hat{\mathbf{x}}(t_i^+) \tag{2.106}$$

The nominal state estimates $\overline{\mathbf{x}}(t_i)$ for position, velocity and attitude are determined using the strapdown propagation equations, depicted in Equations 2.23 and 2.24. The error state propagation equations are determined from Equations 2.23 and 2.24 using perturbation analysis methods; details on this process may be found in [14] and [25].

After filter state initialization, images captured by the cameras are analyzed for SIFT features as described in Section 2.9.1. Next, the nominal trajectory is propagated until the next image capture time using the navigation equations presented in Section 2.4 and information provided by the IMU. A new set of images is captured, and the change in navigation state over this period of time is then used to stochastically project features onto the new images. This projection is used to constrain the search for feature matches between corresponding image pairs. Once features are matched between images, the errors between the predicted and detected feature locations are used to correct the navigation state, and the process repeated. The algorithm is depicted in Figure 2.10. This particular system appears promising as navigation algorithm, yet still exhibits divergent behavior seemingly inherent to EKF based solutions. It is for these reasons that it was chosen as the platform for this research.

Computational and storage savings are found by limiting the number of landmarks tracked by the filter. This restriction is enforced by setting a limit on the amount of time a landmark feature is allowed to remain inactive, meaning that the feature is tracked through time but no corresponding feature is found within newly collected images. If a feature is not successfully matched within this period of time, it is removed from the state vector and replaced with a new feature.

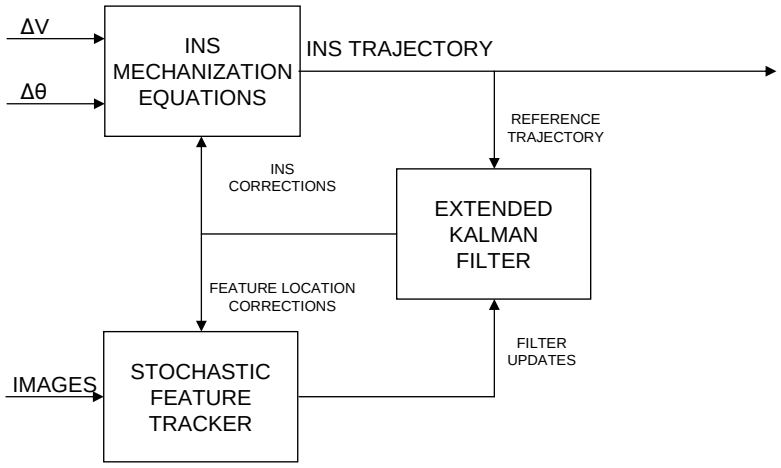

Figure 2.10: Veth's image-aided inertial navigation algorithm. The results of the INS mechanization equations are used to constrain the search for keypoint matches, and the errors between predicted keypoint locations and actual locations are used to correct the INS. The trajectory solution is computed within an EKF, based upon inputs from the imaging and inertial sensors [25].

While [23], [24], and [25] do not address error sources and manifestations inherent to IAN as a central premise, these works provide the foundation upon which the IAN algorithm used in this work is built. Proper characterization of the filter errors relies first and foremost on proper use of the filter, which motivates their inclusion in this section.

2.10.2 Julier and Uhlmann. In Julier and Uhlmann's publication *A Counter Example to the Theory of Simultaneous Localization and Map Building* [7], the authors examine the behavior of a vehicle employing a full-covariance SLAM algorithm to perform localization. The vehicle possesses a sensor capable of measuring the range and bearing of stationary features with respect to the vehicle location in a 2D environment. The vehicle is assumed to have a non-zero initial orientation uncertainty.

The initial investigation within the paper considers the case of a stationary vehicle with no process noise acting upon it. The authors show that the state estimate of the

stationary vehicle will not change over time only if the Kalman gain applied to the vehicle estimate satisfies

$$\nabla h_k^x - \nabla \mathrm{h}_k^p \nabla \mathrm{g}_k^x = \mathbf{0} \tag{2.107}$$

In Equation 2.107, $\nabla h_k^x - \nabla h_k^p$ is the observation model Jacobian and ∇g_k^x is the Jacobian of the inverse of the observation model equation. It is then shown analytically that the gain applied to the state position estimate is indeed correct, but that applied to the orientation estimate is not guaranteed. Proceeding with a simulation, they show that for a stationary vehicle and feature, the position estimate does not change. However, the orientation estimate does change spuriously, and the associated covariance is dramatically reduced, leading to an inconsistent filter. The simulation is then extended to a scenario in which the vehicle is moving, and it is shown that both the vehicle position and orientation estimates become inconsistent. Additionally, the time required for the inconsistent behavior to become evident is on the order of thousands of timesteps.

2.10.3 Bailey, Nieto, Guivant, Stevens and Nebot. The paper *Consistency of the EKF-SLAM Algorithm* [1] explores the symptoms and causes of EKF filter optimism in non-linear SLAM implementations, illustrated through experiments with a robot equipped with a range-bearing sensor that moves through a two-dimensional environment. The authors maintain that inconsistencies are difficult to detect without truth comparison, though may exhibit themselves in larger than expected updates in position and orientation. They observe that if heading uncertainty is kept small through methods such as regular updates from an IMU, then the EKF algorithm exhibits relatively consistent behavior.

The authors first point out glaring problems with the EKF algorithm in their introduction:

1. The state uncertainty mean and variance is based on the assumption of Gaussian system noise, and these moments are linearized, introducing errors

2. The actual distribution is non-Gaussian, so that even if the errors introduced through linearization did not occur, the true mean and variance still do not accurately represent the distribution

It is believed that these are the factors that contribute to filter inconsistency, primarily exhibited as a reduction of the covariance below the true covariance, and jumps in the estimated means of the systems states, including both the vehicle position and the stationary landmarks. To illustrate the first, they consider two experimental setups: one with a stationary vehicle, and the second with the vehicle moving in a rectangular pattern.

In the first setup, the vehicle is initialized with an uncertainty in position and heading $\mathbf{P}_{0|0} = diag(\sigma_x^2, \sigma_y^2, \sigma_\phi^2)$. They note that the true uncertainty of a landmark cannot fall below the initial uncertainty of the vehicle. As the algorithm propagates, the uncertainty of the vehicle position remains constant, while the uncertainty in vehicle heading decreases rapidly. A larger initial uncertainty in heading leads to a greater rate of decrease along with a lower final value in uncertainty estimate. A greater observation uncertainty $\mathbf{R}_k$ also leads to this effect, but to a lesser degree. The gain in information is independent of vehicle position uncertainty $\sigma_x^2$and σ_y^2. They thus conclude that finite $\mathbf{R}_k$ and σ_θ^2 values always lead to an inconsistent EKF update.

For the moving vehicle example, they compare the results obtained from the EKF with those obtained by linearizing the Jacobians about the true states rather than the estimates. The results showed that the covariance estimates for the standard EKF were smaller than those obtained from the ideal Jacobians, thus indicating optimistic information gain. Information gain is negligible if the heading uncertainty is kept below 1.7 degrees. The authors conclude that direct, periodic observation of the vehicle heading such that the uncertainty is kept small will result in a temporarily consistent EKF. Finally, the authors calculate another measure of filter performance, the *average* normalized

estimation error squared (ANEES)

$$\eta(t_i)_{\text{avg}} = \frac{1}{\text{N}} \sum_{\text{k=1}}^{\text{N}} \eta(t_i, k) \tag{2.108}$$

where $\eta(t_i, k)$ is calculated as in Equation 2.66. Using this metric, the authors judge the performance of various filter implementations (EKF, IEKF, UKF, EKF ideal) by comparing the average NEES for each over N Monte Carlo simulations. All became optimistic; even with the heading uncertainty being kept small as described above, the EKF still became optimistic around 4000 seconds.

2.10.4 Taylor. The paper *Improved Fusion of Visual Measurements Through Explicit Modeling of Outliers* [20] by Taylor focuses upon the uncertainty estimate of transforms produced by visual processing algorithms. The thrust of the argument is that feature matches produced by these algorithms fall into two categories: *inliers*, constituted by the appropriate matches, and *outliers*, which are false matches. While most applications use some method of rejecting outliers before computing the frame-to-frame transform, some of these inevitably slip through and in these cases cause an inaccurate transform to be computed. In addition, the presence of outliers tends to result in an inaccurate uncertainty estimate as the outliers are assumed Gaussian. Without an accurate uncertainty, fusion of the output of the image processing algorithm with other values such as inertial measurements is impossible to do correctly.

The author leaves aside the goal of improving the rejection of outliers, and instead focuses on the proper characterization and modeling of the effects of the outliers. To this end he proposes *Outlier-Aware Filtering*, in which the distribution of features is described as as sum of a Gaussian inlier distribution and a uniform outlier distribution

$$PDF = \alpha \frac{1}{V} + \sum_{i=1}^{N_{in}} k_i N(\mu_i, \Sigma_i) \tag{2.109}$$

Using this distribution, a Bayesian filter is developed which computes a sum of multiple probability distributions. As these increase with each time step, low weighted

distributions are removed and their weight added to the outlier distribution. In simulation, it is shown that this approach provides provides a more consistent filter than a statistically gated Kalman filter. The comparison is performed by calculating the *discrete entropy* of each resulting distribution and comparing it with truth

$$entropy = -\sum_{i=1}^{1000000} p_i \log_2 p_i \tag{2.110}$$

The average entropy resulting from the outlier aware filter differed from the truth by 1.5% after 10,000 timesteps.

These results are important to the current work for two reasons. The first is that it provokes awareness of the fact that IMU drift and linearization inaccuracies are not the only possible drivers of poor filter results. The algorithm implemented in this paper uses in fact a form of statistical gating, and makes no concessions for the presence of outliers in the feature distribution. As such, the fusion of the visual and inertial measurements is improperly performed, and the effects will spill over into the final results. The other issue of importance is that a collection of real world data is required to verify the improved performance and uncertainty estimates of the outlier aware filter. This research will provide, as a secondary product, the data required such that this endeavor could be taken on as a future research goal.

2.11 Summary

The contents of this chapter provide the mathematical foundation necessary for the successful completion of the experiment and analysis, which are described in the remainder of the document. The notation and reference frames used within this work were explained. In addition, overviews of coordinate transformations, Kalman filtering, camera calibration, covariance analysis theory, and inertial and image-aided navigation concepts

were also provided. With this information as a basis, Chapter 3 can now be presented, which describes the experimental methods employed during this research endeavor.

3 Methodology

This chapter sets forth the details of the experimental methods employed in the investigation of IAN error characteristics. Section 3.1 describes the data collection platform and associated hardware. Section 3.2 discusses the environment in which the experiments were performed, the path navigated during the collection of the data sets, and the types of data collected. Finally, Section 3.3 concerns the process of compiling the data and the general statistical analysis techniques used.

3.1 Data Collection Platform Description

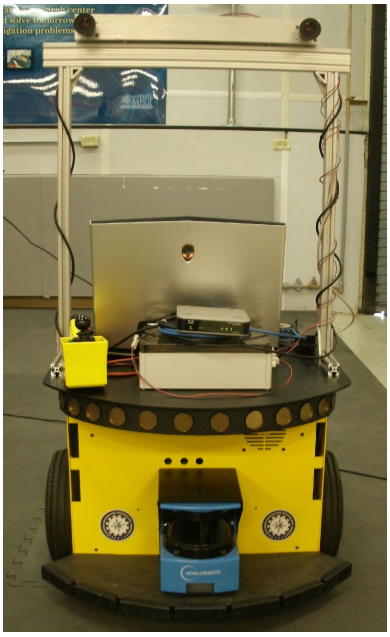

Figure 3.1: The data collection platform. The vehicle is a Powerbot model robot. A metal frame is attached, and the cameras are affixed at each corner of the frame. The IMU is located on the frame, centered between the cameras. The laptop resting on the robot is used to store the collected data.

The data collection platform is built upon the PowerBot model from Adept Mobile Robots. Attached to the top surface of the robot is an adjustable aluminum frame upon

which two Prosilica GC1290 cameras and a Microbotics MIDG II MEMS-grade IMU are mounted. The IMU is mounted near the center of rotation of the platform; the center of the IMU is used as the body frame origin. The cameras are mounted to the left and right of the IMU, approximately 225 mm away along the y-axis of the body frame. An Alienware laptop is used to drive the sensors and handle the data collection, and the platform is manually guided over the course of each run by joystick.

3.2 Data Collection Process

Using the collection platform described in Section 3.1, inertial and image data were collected for 100 runs. The path was rectangular, approximately 30 by 40 meters. The collection platform was guided by joystick down the center of each hallway until a specified point was reached, and speed was kept as uniform as possible. The platform then performed a 90 degree left-handed turn and continued down the hallway. In addition, the platform was brought to a complete stop at each corner; turns were conducted in the absence of intentional translational motion.

The environment in which the data was collected was controlled for multiple variables. Areas that were monitored included: lighting conditions in the hallway; the disposition of commonly found features, namely trashcans, billboards, and doors; and pedestrian traffic. The purpose of this monitoring was to ensure to the greatest extent possible that the features tracked within the images would be consistent across runs. Collection was performed during the early morning and in the evening to avoid pedestrians entering the camera field of view or altering the scene between runs. Either occurrence could result in tracked features being obscured and subsequently dropped. Although such restrictions are incompatible with what would be encountered in a real-world scenario, uniformity in the collection environment over time simplifies the process of isolating other irregularities in filter operation for analysis.

A full 119 runs were collected, with 19 runs being deemed unusable. The rejected runs all contained one or more violations of the controlled conditions occurring during collection, such as lights being turned on or off, the placing of new objects in the hallway, or pedestrians loitering in view of the cameras. At the beginning of each run, the platform was allowed to remain stationary at the starting location for 90 seconds. The IMU output for this period of time was provided to the navigation filter along with the true position, velocity and heading of the stationary platform. This enabled the navigation filter to estimate the accelerometer and gyro biases in the IMU.

For all collections, the cameras were triggered to collect images at a rate of 2 Hz, while the IMU provided acceleration and turn rate data at 50 Hz. The camera images were generally timestamped from the IMU time output; in the case of a communication lag between the IMU and computer, the images were instead stamped with PC time from the laptop. In addition to the inertial and imaging data for each run, images of the checkered calibration board were also captured immediately after each collection session. Provided with stereo images, the dimensions of the calibration board and the pixel locations of the board corners in each image, the CCCT software can calculate the intrinsic and extrinsic parameters for each session. This accounted for any changes that occurred due to accidental jostling of the rig or tampering with the cameras. At the end of each collection, all data was transferred to a PC for processing.

3.3 Data Processing

Once the inertial and imaging data were collected, the data underwent pre-processing to make it compatible with the navigation filter software. First, runs that were deemed unsuitable during the collection phased were removed from the primary data pool and set aside, but not discarded. Next, computation of the camera intrinsic and extrinsic parameters was performed using the Caltech Camera Calibration Toolbox for MATLAB®. Then the image filenames were changed into a standardized format that

included the name of the camera that collected the image, the GPS week, and MIDG IMU timestamp data. Alignment of the image files followed, with the images being checked for matching timestamps. If an image did not have a match from the other camera, it was removed from the primary image set and put aside. The inertial data from the INS was then converted from binary to ASCII format that could be stored in a MATLAB® array. The next step was the determination of the truth for comparison points, and this procedure was performed as follows.

Coordinates for points along the hallway path were obtained through a surveying process completed before the advent of this research, and these were used as position reference points. Assumptions were then made concerning the nature of the platform movement as follows: when navigating down a hallway, the platform moves at a constant speed in a straight line connecting one surveyed corner location to the next, and during the execution of a turn, the turn rate is constant and the platform undergoes no translational motion. These assumptions, along with the knowledge that the path consists of straight segments connected by 90 deg turns, allowed for the 'filling in', through interpolation, of the trajectory points that lay between the surveyed data points.

The final step in preparing the data for the navigation filter was image rectification using the previously determined camera calibration parameters and the extraction of SIFT keypoints for each image. This step was performed outside of the filter operation to reduce the runtime of the filter algorithm. The prepared data was then fed to the navigation EKF software, which uses the inertial data and images to estimate the trajectory of the platform over time. Once the filter is finished processing a given run, the estimated position, velocity and heading of the collection platform is saved in a unique file, along with the the truth and the values of various filter settings described in the next section.

3.4 Models

Proper functionality of the navigation filter requires that certain characteristics of the associated hardware be known and available to the filter. Examples include the location of the cameras relative to the IMU, and the properties of the IMU itself. In addition, there are a number of variable parameters within the algorithm that must be set. This section reviews the various types of prior information that must be provided to the filter, and reports the values used within this work, where appropriate.

3.4.1 Navigation EKF Variables. The navigation filter contains multiple parameters that may be adjusted to fit a given run scenario, add functionality or reduce computational burden and memory requirements. The most pertinent of these are described as follows.

Coast Time: The amount of time a SIFT feature will be maintained by the filter without being successfully matched before it is dropped.

Max Tracked Landmarks: This is the maximum number of SIFT features tracked at any given moment.

Max Landmarks Added: This is the maximum number of new SIFT features allowed to be added at a given timestep.

Scale Pixel Error Coefficient: This value modifies the uncertainty of a feature measurement as a function of the feature scale, with larger features having a greater uncertainty in pixel location.

Radial Pixel Error Coefficient: This value increases the uncertainty of a pixel measurement as a function of radial distance from the center of the image to compensate for the effects of unmodeled radial distortion.

Pixel Measurement Uncertainty: Standard deviation associated with noise in image measurements.

SIFT Match Threshold: The correlation between a candidate match and a reference feature must be greater than this value. Based upon the magnitude of the dot product between a candidate feature and a base feature.

SIFT Distinction Threshold: The ratio of the similarities between two candidate matches when compared to a reference feature.

Stochastic Constraint Ellipses: Base length of the ellipse which is to be searched for a feature match.

Max Propagation Timestep: Incremental timestep within the filter over which measurements are integrated. This is applied if the time until the next update is greater than the max propagation timestep in order to force an update in the covariance. Otherwise, the filter propagates until the next update event.

The values of each parameter were kept constant over the course of this research, and are summarized in Table 3.1.

3.4.2 Camera Models. During the course of this research, the cameras and lenses used were not changed, nor were their locations relative the collection platform and inertial sensor. However, as the data collection phase of this worked spanned weeks, there existed the potential for the cameras to be disturbed in such a manner as to subtly alter key characteristics, such as focal length. To account for possible changes, camera calibration was performed for each collection session. A representative selection of camera parameters is shown in Table 3.2. A full account of the variable camera parameters for each collection session is included in Appendix A, in Tables A.1, A.2, and A.3.

Table 3.1: IAEKF parameters. The table depicts the parameter values used in the IAEKF algorithm throughout this work.

Coast Time (s)	1
Max Tracked Landmarks	20
Max Landmarks Added	10
Scale Pixel Error Coefficient (pix)	0.05
Radial Pixel Error Coefficient (pix)	0.005
Pixel Measurement Uncertainty (pix)	2
SIFT Match Threshold	0.95
SIFT Distinction Threshold	0.45
Stochastic Constraint Ellipses (pix)	2.15
Propagation Timestep (s)	0.1

Table 3.2: Sample Camera Calibration Parameters for Run 1. Camera 125495 was the left camera in the rig, while 122865 was the right. The values presented were obtained from the CCCT software.

	125495	**122865**
Resolution (pix)	1280×960	1280×960
Focal Length (mm)	1354.9, 1354.9	1355.5, 1356.1
Principal Point (pix)	667.95, 537.01	602.25, 504.70
k_1	-0.12604	-0.13914
k_2	0.15176	0.22382
k_3	-0.00017	0.00059
k_4	0.00005	-0.00032

3.4.3 IMU Model. The IMU model used in this research is a MIDG II

MEMS-grade INS, identical to that which was used by during the course of Jurado's work

[8], [9]. As such, this work uses the model parameters cited in [9], which are reproduced in Table 3.3 for ease of reference.

Table 3.3: MIDG model parameters. Reproduced from [9].

Sampling frequency (Hz)	50
Accelerometer time constant (s)	3600
Gyroscope time constant (s)	3600
Accelerometer bias STD (m/s^2)	0.2
Gyroscope bias STD (rad/s)	9×10^{-3}
Accelerometer scale factor (ppm)	300
Gyroscope scale factor (ppm)	150
Accelerometer random walk STD (m/s/ $\sqrt{\text{s}}$)	9×10^{-3}
Gyroscope random walk STD (rad/ $\sqrt{\text{s}}$)	9×10^{-4}

3.4.4 Sensor Installation. In a manner similar to the camera intrinsic parameters, the extrinsic parameters were re-calculated for each collection session. The Caltech Camera Calibration Toolbox is capable of performing a computation of the relative translation and rotation of the right side camera with respect to the left camera, and returns these as vectors **T** and **om**, respectively. These are given in the Caltech frame, described in Section 2.2. The vector **om** describes an axis about which a pointing vector is to be rotated, and the magnitude of **om** gives the measure of the angle through which the rotation occurs. Vector **om** is related to the rotational DCM through the Rodrigues equation [3][5]:

$$\mathbf{C}_{\text{a}}^{\text{b}} = \mathbf{I} + (\sin\Theta)\widetilde{\mathbf{om}}_{\times} + (1-\cos\Theta)\widetilde{\mathbf{om}}_{\times}^{2} \tag{3.1}$$

where $\widetilde{\mathbf{om}}$ represents the unit vector in the direction of **om**, and the subscript $\times$ denotes the skew-symmetric matrix form. The angle Θ is the angle about $\widetilde{\mathbf{om}}$ through

which rotation occurs. Combined with the knowledge that the cameras are placed equidistant from the IMU in opposite directions along the y_b-axis, the relative rotation and translation between each camera frame c and the b-frame can be determined.

3.5 Summary

This chapter provided an overview of the processes used in collecting the data for analysis, the truth against which the data was compared, and the preparation of the data for use in the filter. Also presented were various parameters that were required for filter operation. Chapter 4 contains the details of the results of the collection process, and the analysis of those results.

4 Results and Analysis

Presented in this chapter is a summary of the data collected during the process outlined in Section 3.2, the methods employed in the analysis of the data, and the resulting plots and statistics. The chapter concludes with a summary of the most notable results.

4.1 Overview of Data Collection

As stated in Section 3.2, the goal of the experimental process was to obtain 100 sets of stereo images and inertial measurements, collected as the platform navigated an indoor path in a controlled environment. The similarity of the runs allows Monte Carlo analysis methods to be applied.

The data collection took place over an approximately 11 week period beginning 26 June, 2012. As the experiment was conducted within an academic facility, maintaining control of the environmental factors was a particular challenge. In order to reduce the frequency of situations which would result in an invalid run, all collections took place outside of the normal workday, typically in the periods of 0500-0700 and 1800-2000.

Despite the precautionary measures, on occasion an alteration the environment such as camera occlusion by a bystander or a change in the lighting conditions compromised the integrity of the collection. As a result of these occurrences, a total of 119 data sets were collected over this period. Of these, 19 were deemed unsuitably similar in the scene observed to allow for proper comparison with the other runs, and these data sets were put aside. Once each collection session had ended, all of the data sets from that particular session were transferred to a hard drive and moved to a desktop workstation, where they underwent pre-processing according to the methods described in Section 3.3. Once the formatting and calibration steps were completed, the data was then processed through the IAN algorithm to obtain position, velocity, and attitude estimates along with associated covariances.

On average, the length of each run was 237.7 seconds, not including the 90 second alignment time preceding each collection. The variation in the length of time required to complete each run was due to small deviations from the path introduced by manual control of the robot. Due to the differences in the length of each run, it was not possible to simply compare each run on the basis of the original time indices. Doing so would result in comparison of state errors at differing points along the path. To alleviate this issue, the filter output for each run was divided into eight sections: the alignment period, the four straight segments of the path, and the three turns. The time vector corresponding to each section was expanded, and the filter output for section was then linearly interpolated in time over the new time vector. This ensured that for all runs, each segment of the filter output consisted of the same number of data points without a loss of information. The interpolated data could then be compared on the basis of the new time indices. This process was also applied to the truth source for each run. Table 4.1 shows the average length of time over all runs, and the length of the interpolated data. It should be noted that the filter produced output records at a rate of 2 Hz, thus the interpolated output for all runs consists of a total of 712 data points for each data type. In all remaining analysis, the runs are compared on the basis of, and plotted against, the interpolated time indices i.

The estimated North vs. East trajectory, when plotted against truth, may be used to compare the performance of the filter during each run. The N vs. E trajectory plots for all 100 runs are shown in Figures 4.1, 4.2, 4.3, 4.4, and 4.5, with the red line representing the truth path, the blue representing the filter estimate for the given run, the small circle denoting the starting location of the platform, and the arrow the direction of travel.

Table 4.1: Comparison of original and interpolated data lengths. Original length of filter output, in seconds (s), and length of interpolated output (i). There are two filter outputs for every second s in the original data. The time index units i of the interpolated data refer directly to output records; thus only integer values are used. The interpolated data lengths were chosen to be slightly longer than the longest corresponding number of data points seen within the 100 run set.

Path Section	**Original Average (s)**	**Interpolated** (i)
Alignment	90.255	200
Path 1	45.867	100
Turn 1	2.882	8
Path 2	67.391	140
Turn 2	2.811	8
Path 3	45.653	100
Turn 3	2.790	8
Path 4	67.387	140
Turn 4	2.884	8
Full path (includes alignment)	327.920	712

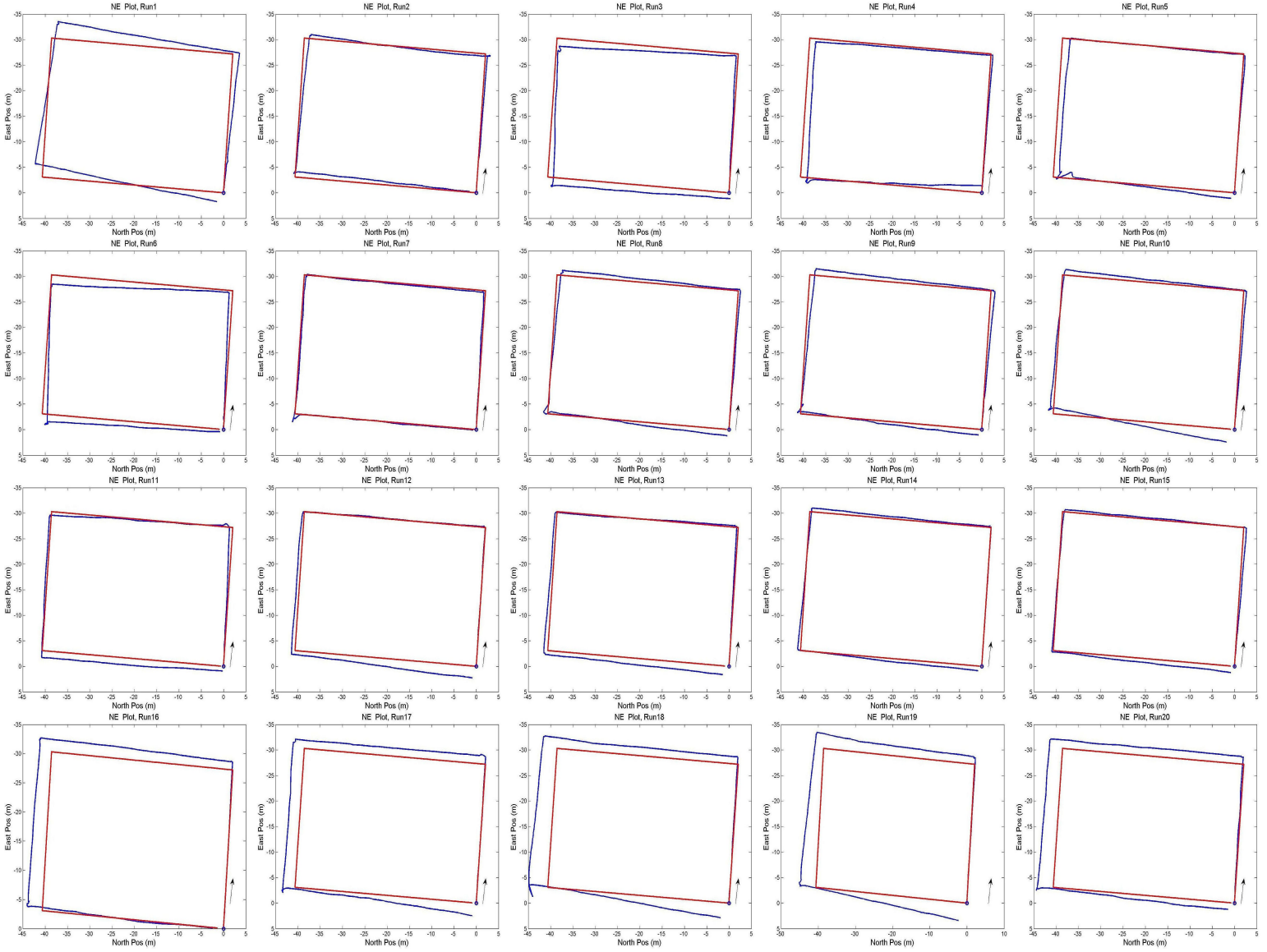

Figure 4.1: N vs. E Position Estimate, Runs 1 Through 20

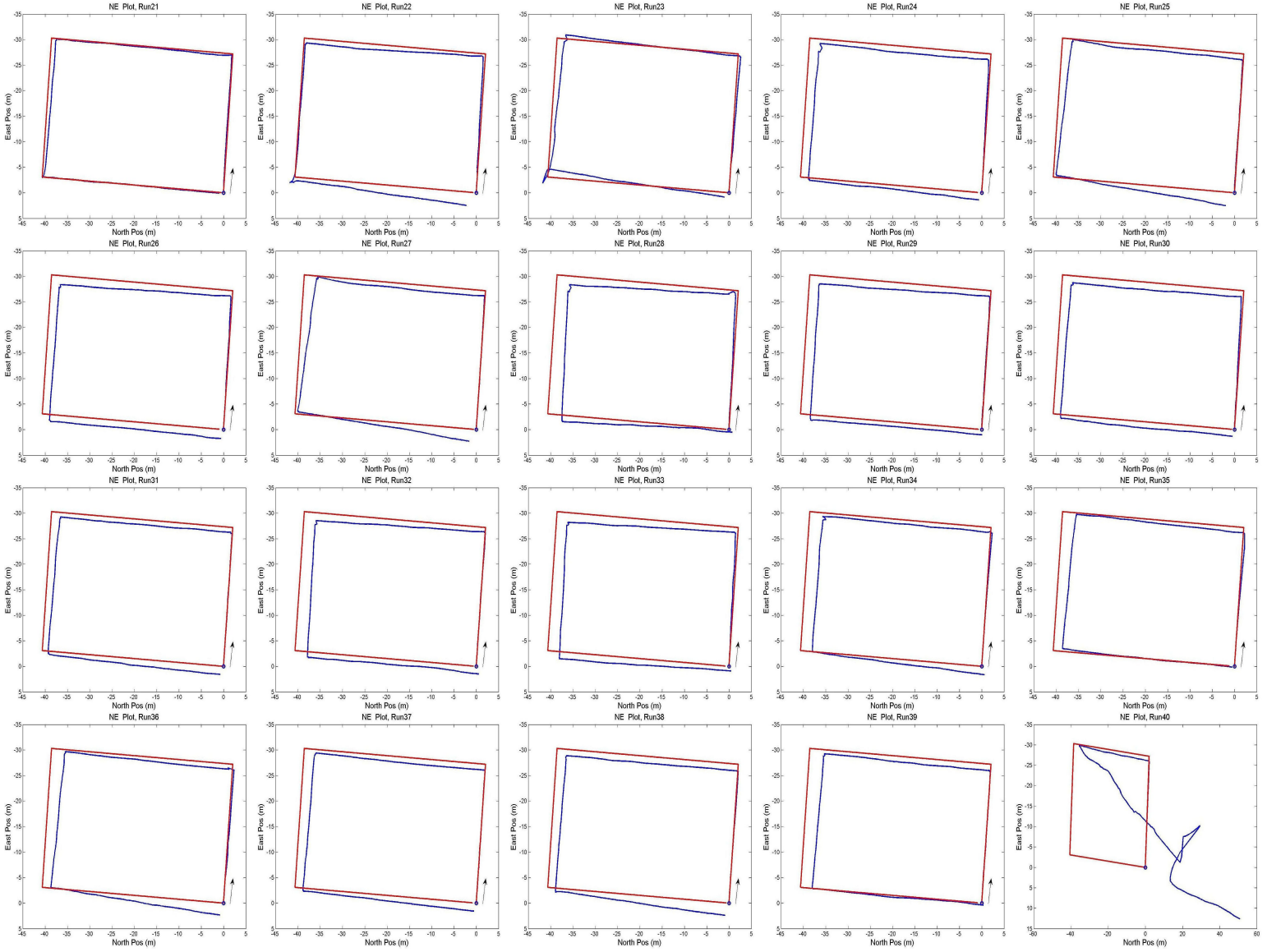

Figure 4.2: N vs. E Position Estimate, Runs 21 Through 40

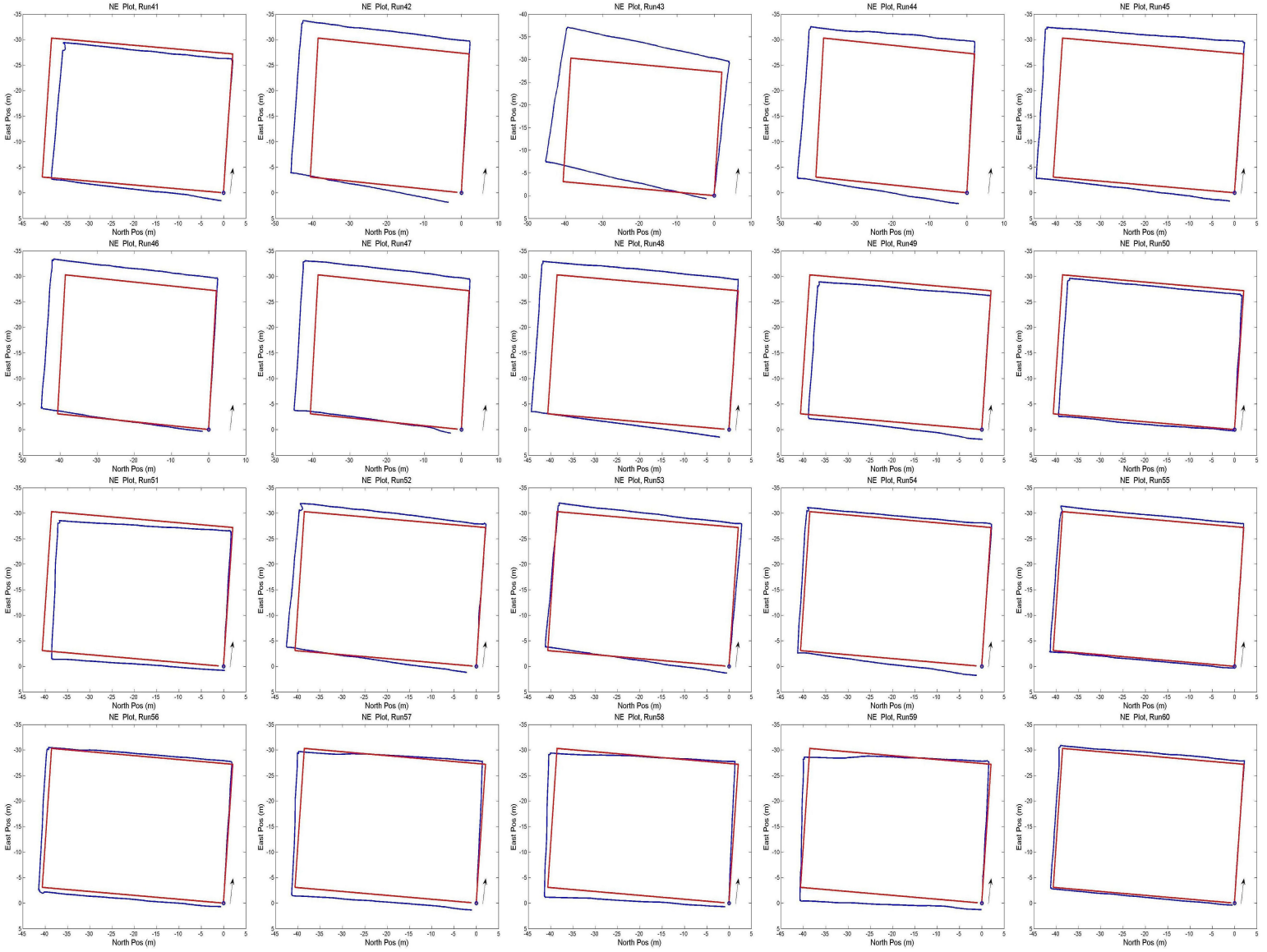

Figure 4.3: N vs. E Position Estimate, Runs 41 Through 60

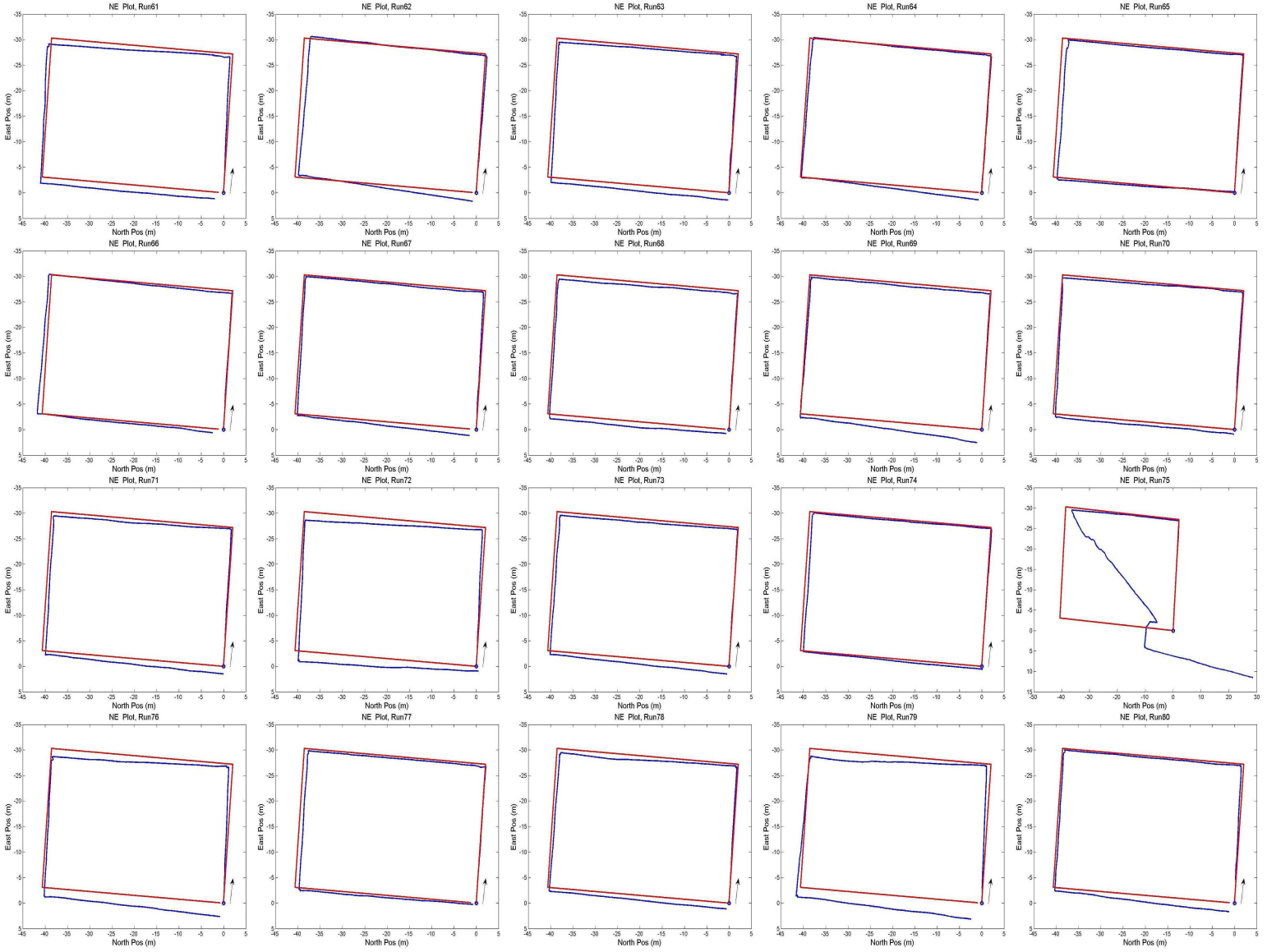

Figure 4.4: N vs. E Position Estimate, Runs 61 Through 80

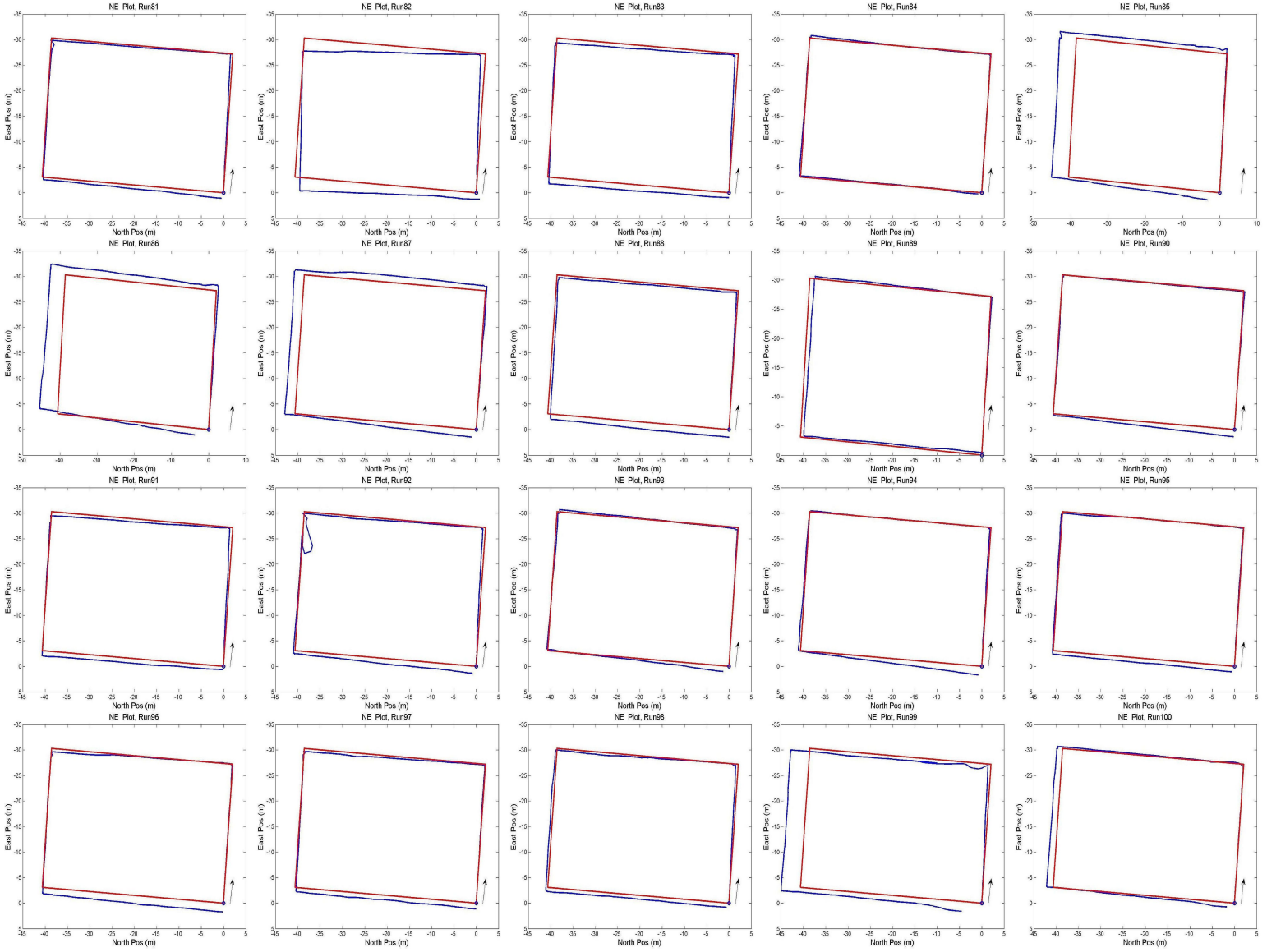

Figure 4.5: N vs. E Position Estimate, Runs 81 Through 100

As is seen in Figures 4.1 through 4.5, for 98% of the collected runs, the filter performs as expected. Errors in the position estimate accrue slowly over time, indicating that the image-aiding is not removing all of the drift associated with the inertial measurements, small errors are being introduced through the feature matching algorithm, or a mixture of the two. However, this degradation in the quality of the estimate over time is in line with simulation results presented in [25], and is thus expected.

Runs 40 and 75 are seen to exhibit a different behavior than the remaining 98 runs. Each of these is observed to estimate the trajectory in a manner consistent with the other runs for the first half of the path; however, as the platform enters the second turn in each run, something causes the filter estimate to diverge catastrophically. Large errors are seen in the position estimate for the remainder of each run. Further consideration of the divergent runs can be found in Section 4.5. Statistical analysis of the non-divergent runs follows, beginning with a consideration of positioning errors in Section 4.2. It is important to note that the remaining analysis and the associated plots only considers the measurements after $i = 200$, this point being the end of the alignment time, post interpolation.

4.2 Position Errors

4.2.1 Position Estimation Errors. Shown in Figures 4.6, 4.7 and 4.8 are the errors in the position estimate for the North, East and Down directions for the 98 non-divergent runs. The errors for each run were calculated by subtracting the truth from the estimate, conditioned on the run number k and the interpolated time indices i, i.e.,

$$\boldsymbol{\epsilon}_{k,i} = \hat{\mathbf{x}}_{k,i} - \mathbf{x}_{k,i} \tag{4.1}$$

The thick red line shown on each plot is the sample mean of the errors in the given direction, calculated as

$$\mu_i = \frac{1}{n} \sum_{k=1}^{n} \epsilon_{k,i} \tag{4.2}$$

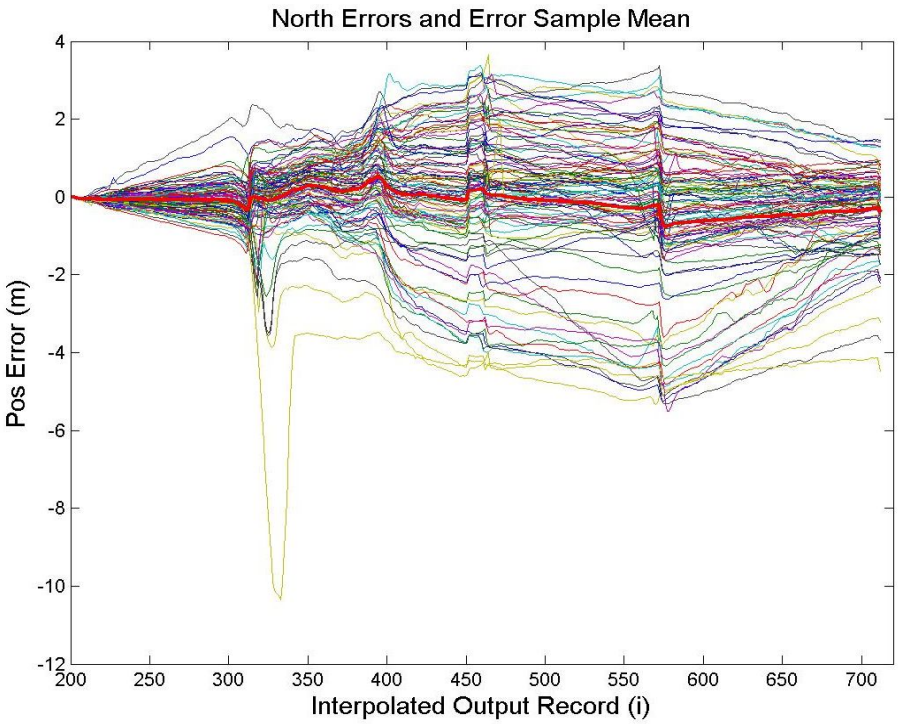

Figure 4.6: North Position Ensemble Errors.

As can be seen in Figures 4.6 and 4.7, neither the North nor East position errors are zero-mean nor uncorrelated in time. Taking the time average of the mean error for both the North and East directions results in an average error over all runs and time for the North direction of -10.4 cm, and 20.7 cm in the East direction. Considering time correlation of the positioning errors, particularly in the areas of $i = [300, 450, 570]$, trends appear that span all runs. These trends are manifested as a brief, sudden increase or decrease in the magnitude of the error, sometimes accompanied by a plateau, depending on the time and what direction is being considered. Each of the times at which this behavior occurs corresponds roughly to the times at which a turn is navigated. The 'bumps' in error are partially due to the greater challenge presented to the filter during the navigation of a turn. Rapidly changing angles are more difficult for the gyros to measure,

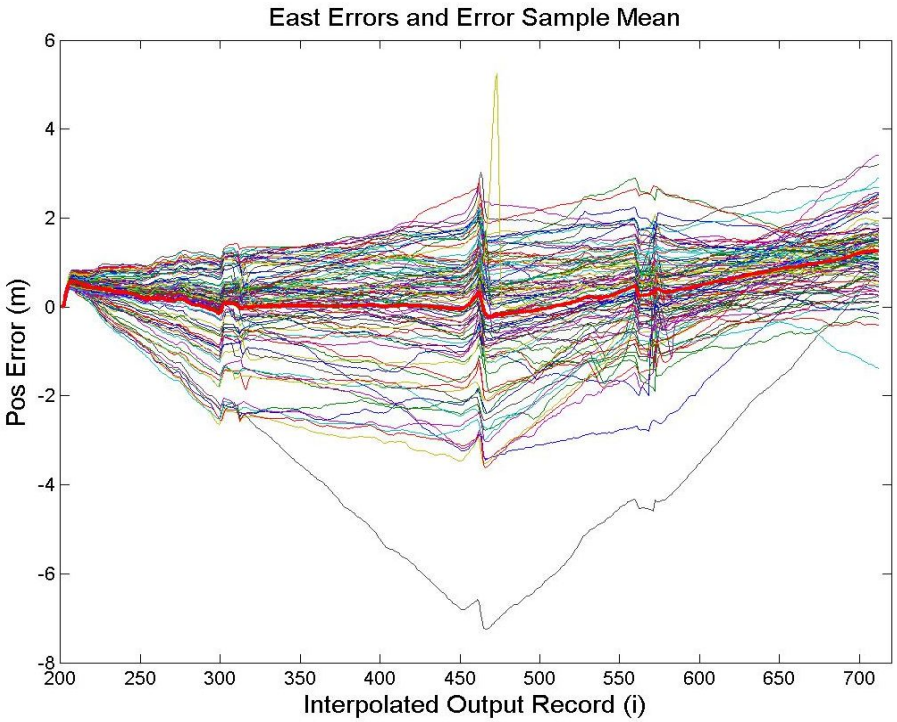

Figure 4.7: East Position Ensemble Errors.

and the scene evolves so quickly during a turn that the filter is unable to match features from one frame to the next, resulting in few or no corrections to the IMU.

Another source of the error bumps seen in Figures 4.6 and 4.7 are the assumptions that went into the truth data. As stated in Section 3.3, a constant velocity is assumed over each straight path. This does not take into consideration any acceleration that may occur entering and leaving turns. Taking each path segment separately, we see that the platform first moves primarily West as it heads to the first turn. The platform slows down before coming to a complete stop and turning; however, the truth assumes a constant velocity, and an increase in error in the East direction is seen over each run in Figure 4.7 near $i = 300$. The platform turns and heads South, slows down again coming into the second turn, and a positive error in seen in the North direction in Figure 4.6 near $i = 450$. At the third turn, occurring at $i = 570$, the behavior is seen again in Figure 4.7, but the error is negative in

magnitude as the platform is traveling in the East direction. These time correlated errors may be introduced due to the poor quality of the truth source about each turn.

The position errors in the Down direction must be discussed separately. As shown in Figure 4.8, there is an obvious bias in the negative Down direction. Similar behavior does not appear in either the North or East directions, as well it should not; the biases in the inertial sensors are estimated by the filter and removed from the estimate. The upward bias that is seen in Figure 4.8 appears to be introduced through the feature matching updates. One possibility is that there is an error in the extrinsic camera parameter measurements, causing a persistent error in the feature depth estimates obtained through the binocular stereopsis calculations detailed in Section 2.9.2.

Upon further consideration of Figure 4.8, position errors are seen to occur in the time frames around each turn in a manner similar to those in Figures 4.6 and 4.7. However, as

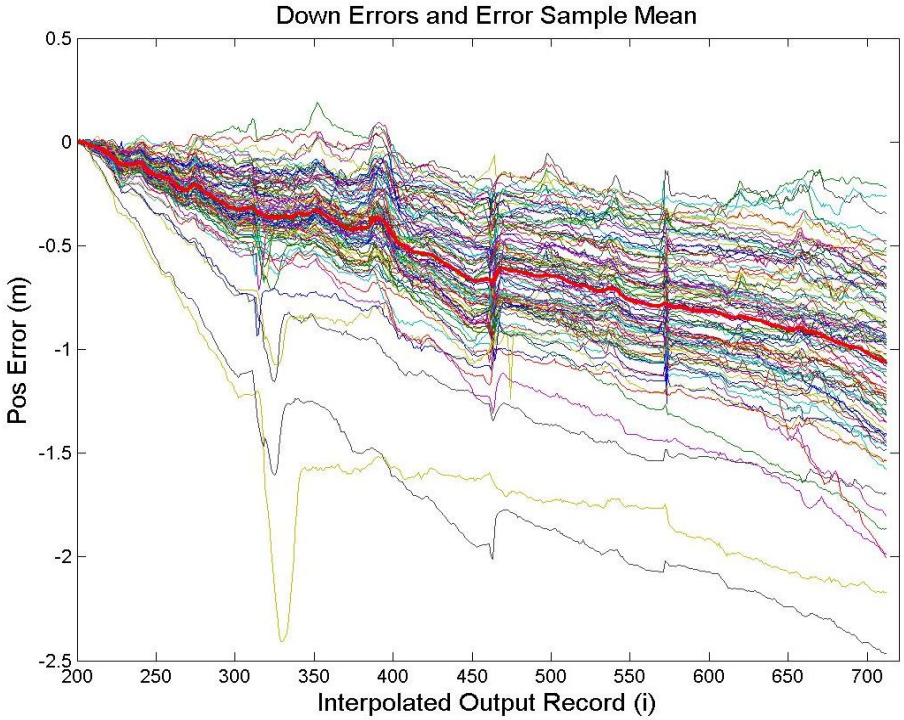

Figure 4.8: Down Position Ensemble Errors.

the floor in the hallway in which the data collection occurred is essentially level, the error bumps cannot be even partially attributed to errors in the truth as previously argued for the North and East directions. As was also stated in the consideration of the North and East positioning errors, the error bumps in the Down direction near the times $i = [300, 450, 570]$ may be attributed to the general increase in difficulty of estimation posed by a turn as compared to a straight path. This may be seen as evidence that the relatively poor quality of the truth in the area around the turns is not the sole or even dominant source of positioning error.

In Figures 4.9, 4.10, and 4.11, the mean error is reproduced, along with the $\pm 1\sigma$ standard deviation bounds computed by taking the square root of the sample covariance equation, or

$$\sigma^2_{i,\text{sample}} = \frac{1}{n}\sum_{i=1}^{n}[\hat{\mathbf{x}}_i - \mu_i][\hat{\mathbf{x}}_i - \mu_i]^{\mathbf{T}} \tag{4.3}$$

In the North direction, the sample standard deviation reaches a peak value of 1.9919 m at i =573. In the East direction, the sample standard deviation peak is 1.5738 m at i = 464. It is seen that the standard deviation decreases in both cases, reaching respective values of 0.9957 m and 0.7785 m at the final data point. This is somewhat expected. The filter has dedicated state variables for estimating the biases in both the accelerometers and gyroscopes. However, the alignment preceding each collection was only performed for 90 s, and it is certain that the estimation of the bias values was imperfect. This remaining bias, combined with the fact that the platform is traversing a closed, rectangular path, can account for the relative reduction in estimations error towards the end of each run, and thus a reduction in the sample standard deviation. Biases that contribute to an error that are added while moving in the South direction, for example, would be removed as the platform navigates in the exact opposite direction along a path of

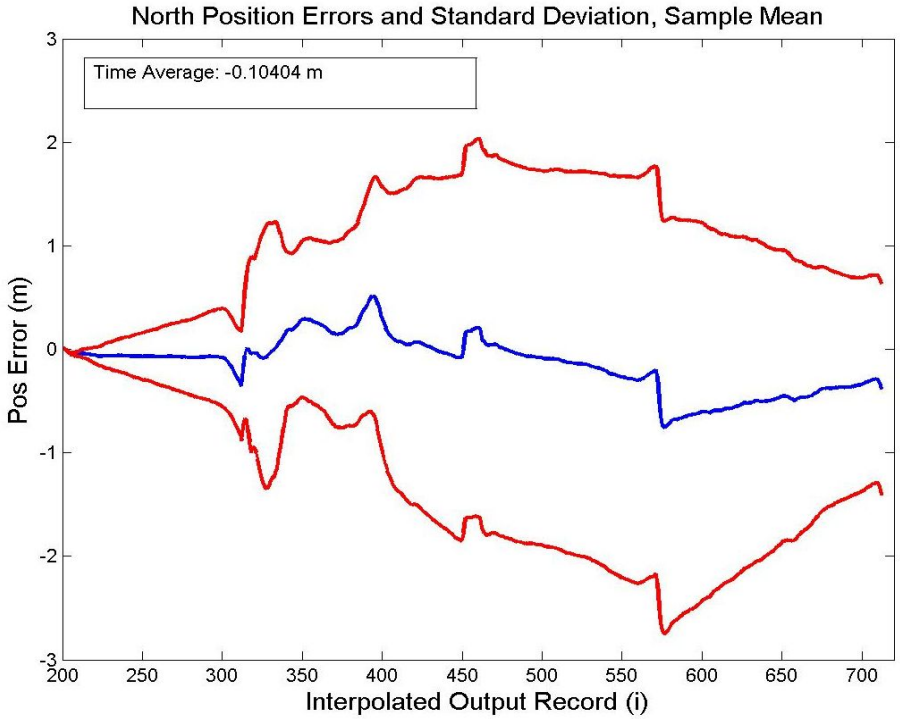

Figure 4.9: North Position Error Sample Mean and Standard Deviation.

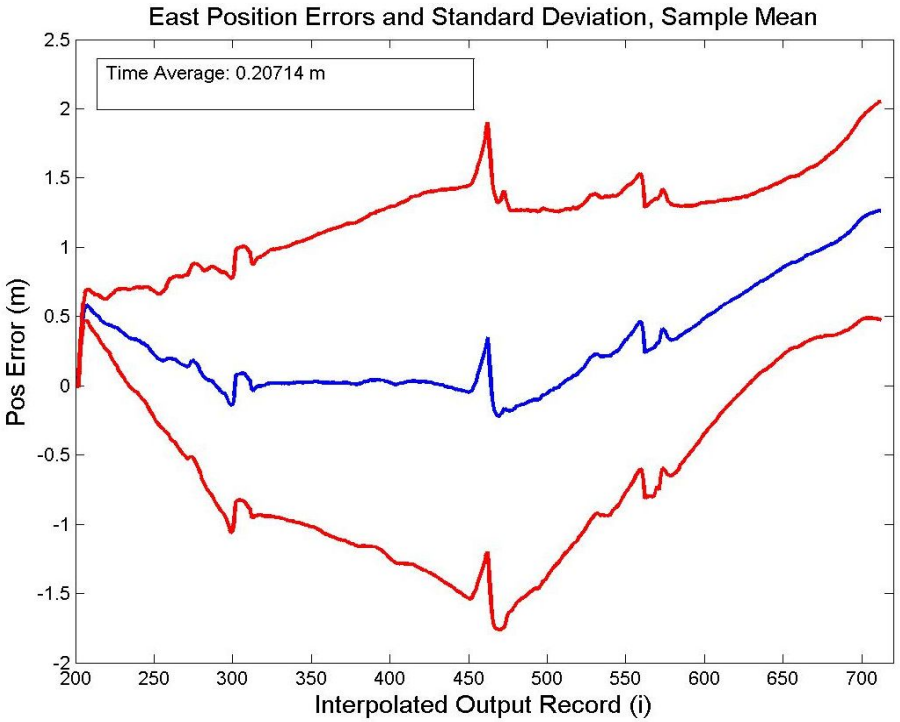

Figure 4.10: East Position Error Sample Mean and Standard Deviation.

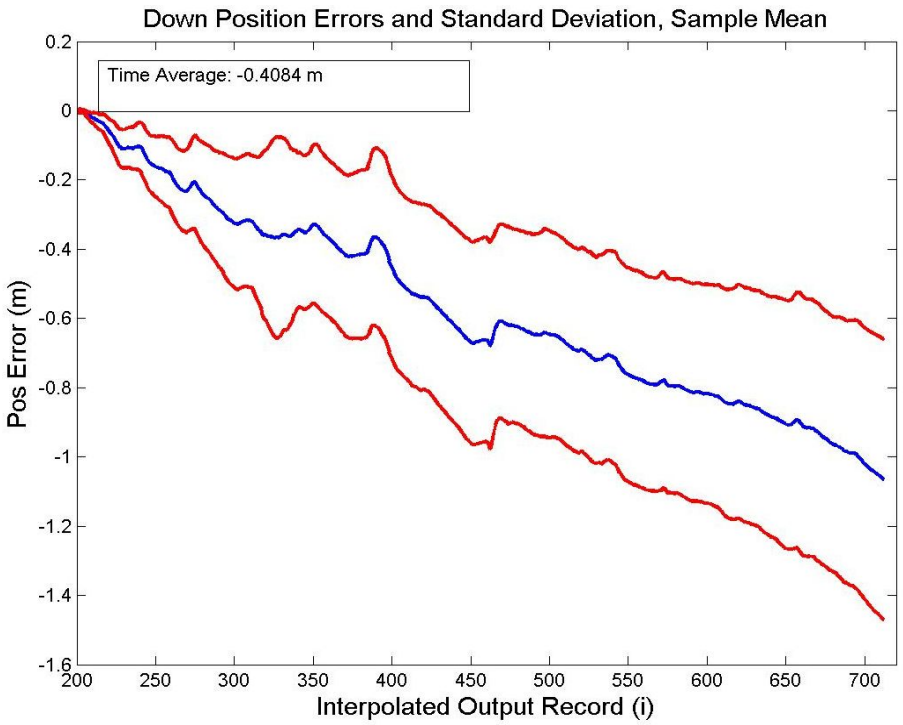

Figure 4.11: Down Position Error Sample Mean and Standard Deviation.

the same length. A similar reduction of errors occurs in connection with the neglect of acceleration entering and leaving each turn.

The standard deviation in the Down direction also shows a dissimilar behavior, as would be expected from the near monotonically increasing error magnitudes and spread about the error mean with respect to time. The standard deviation for the Down direction increases with time, reaching a final peak value of 0.4044 m.

4.2.2 RMSE Values. One metric that is used to described the magnitude of the errors associated with a system is the Root Mean Squared Error (RMSE), which combines the errors seen over an arbitrary number of dimensions and runs into a single, time-varying value. The RMSE is calculated as

$$\text{RMSE}_i = \sqrt{\frac{\epsilon_{i,1}^2 + \epsilon_{i,2}^2 + ... + \epsilon_{i,n}^2}{n}} \tag{4.4}$$

where i is the time index and n is the number of dimensions.

The 3D RMSE for the position estimates are shown in Figure 4.12. The RMSE for each individual run was calculated using Equation 4.4; these values are shown in the upper plot in Figure 4.12. The lower plot shows the average 3D RMSE over all runs. As can be seen, the 3D RMSE was held to below 1.3 m over the entire length of the runs, which is an acceptably small error magnitude for all but the most precise navigation requirements. Also noteworthy is the fact that the average errors decrease after the second and third turns due to the removal of both estimation error associated with the neglect of acceleration in the truth and unestimated bias.

As the path navigated during the course of data collection is known to be relatively level, and the Down position errors seen in Figure 4.8 are seemingly dominated by a spurious negative downward bias and generally smaller in scale the either the North or East estimation errors, it is instructive to examine the 2D RMSE errors as well. These are shown in Figure 4.13. Knowing that navigation is taking place in an area which may be assumed level, the errors in altitude would be of little concern. Removal of the smaller magnitude downward errors has the effect of essentially shifting the 3D RMSE graph upwards. However, the errors are seen to still be rather small, peaking at 1.447 m at $i = 462$. Average RMSE for both the 2D and 3D cases are below 1 m.

4.2.3 Position Error Histograms. Having determined the positioning errors in the n-frame, an idea of the error distribution can be obtained through the production of histograms. Following the method detailed in Section 2.7.1, the position errors in each direction were binned at each timestep to produce error histograms. Samples of these were taken at timesteps $i = [304, 452, 560, 712]$, representing the midway points of each

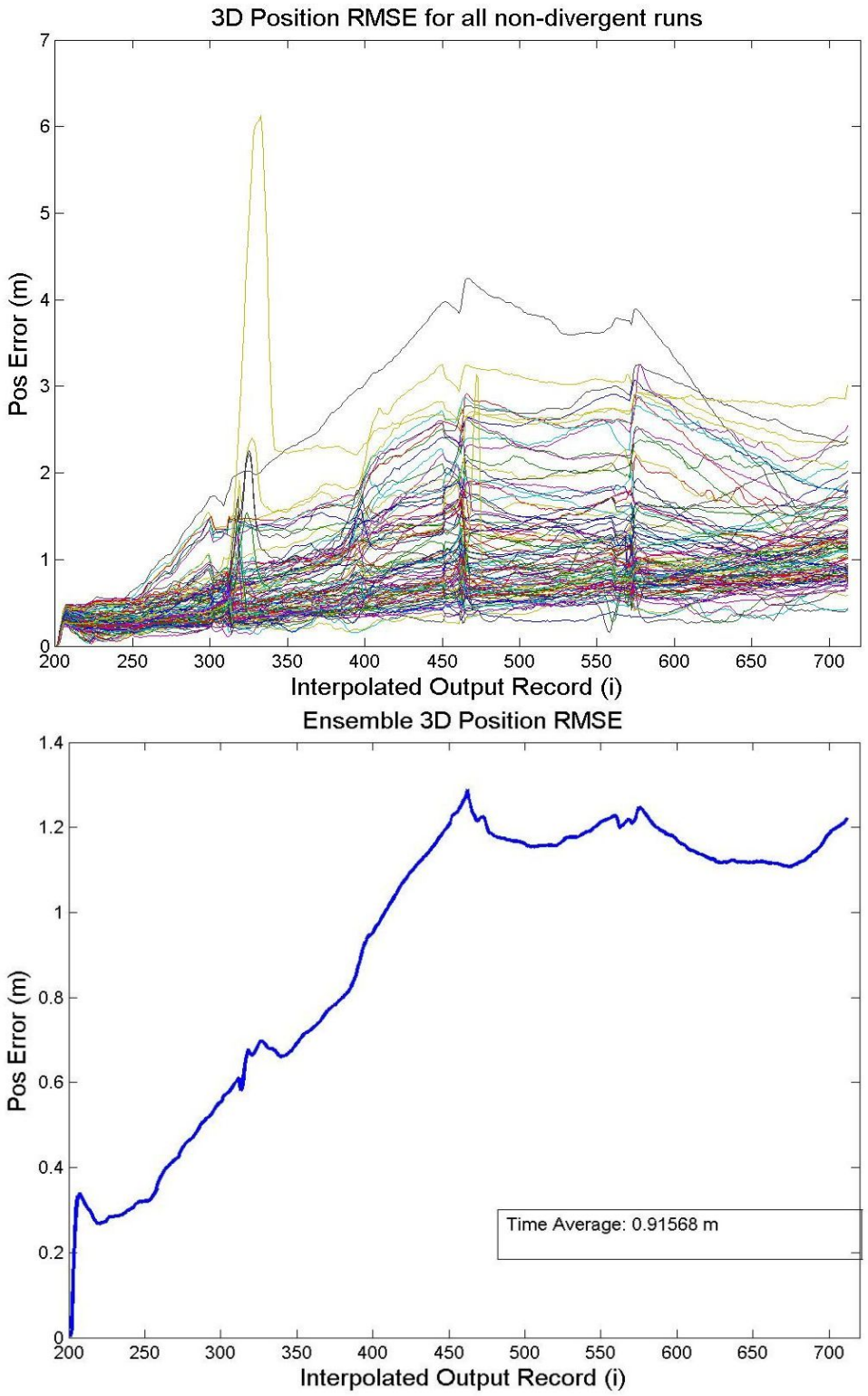

Figure 4.12: 3D RMSE, Position.

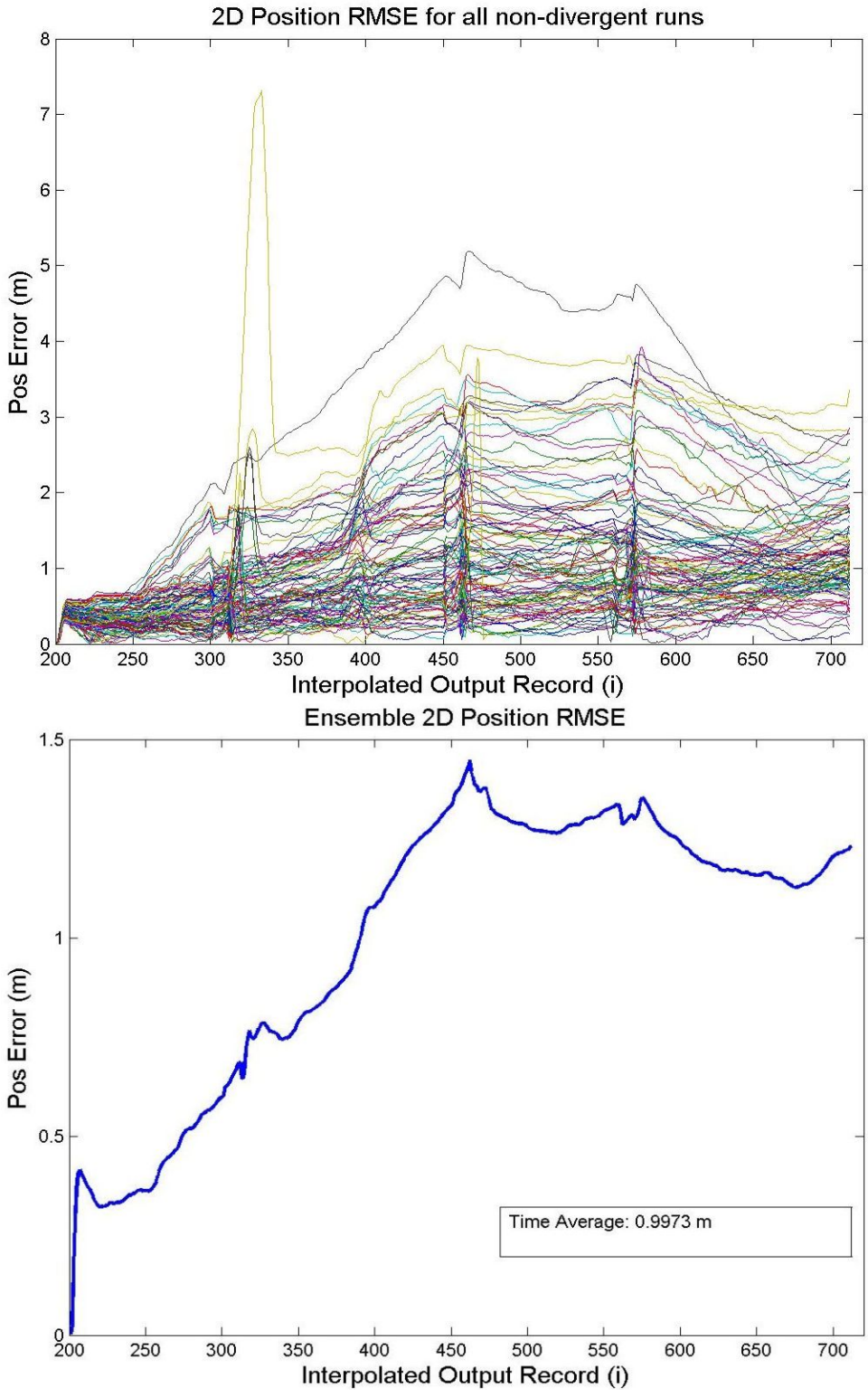

Figure 4.13: 2D RMSE, Position.

of the turns and the endpoint, and are shown in Figures 4.14 through 4.19. The histogram values were normalized such that the sum of all binned values equals 1. Additionally, assuming that the errors can be adequately described as Gaussian, a Gaussian fit to the errors can be determined. This was performed to see how well the error histograms fit this assumption, and the Gaussian curve is overlaid onto its respective histogram. Indicated in the boxes contained within each figure are the μ and 1 σ values of the estimated Gaussian fit. In addition, the normalized histogram values are scaled against the distribution curve for ease of viewing.

Upon viewing Figures 4.14, 4.15, 4.16 and 4.17, it is seen that a Gaussian distribution cannot be assumed for all times. Considering the North position errors at $i = 304$, the distribution appears to be Gaussian with a mean of $\mu = -0.136$ m, with two stray errors outside the distribution. As time continues to $i = 452$ and $i = 560$, the histograms appear to retain a Gaussian core, but a sizable number of errors are seen in the -2 to -5 m range which do not appear to fit a Gaussian distribution. This 'outlier' error grouping, when taken separately from the Gaussian cluster of errors ranging from -2 to 3, appears to have a uniform distribution. In these instances, it is possible that viewing the error distributions as the sum of a Gaussian and uniform distribution is more appropriate, similar to the proposed distribution of feature outliers presented in [20].

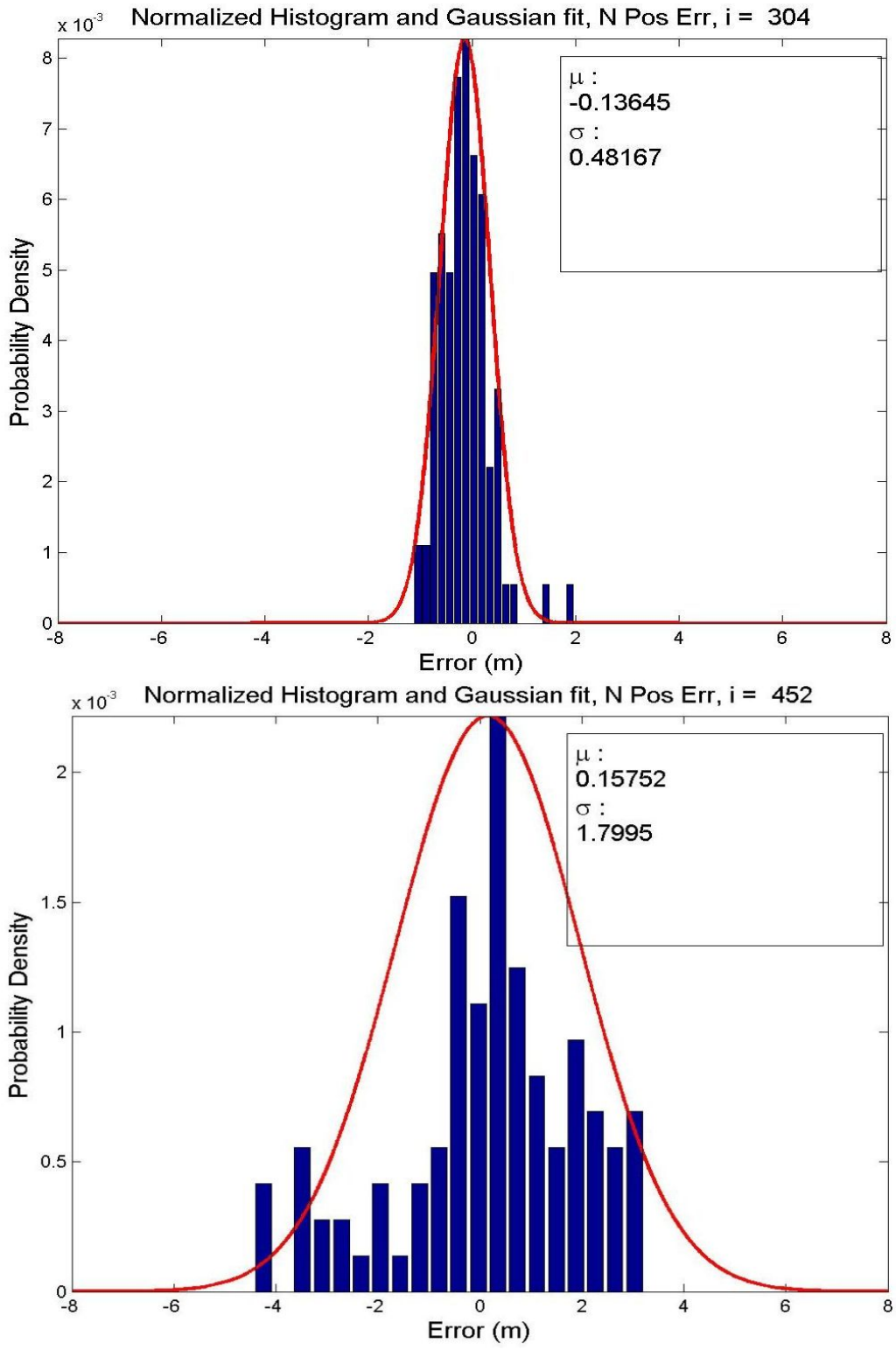

Figure 4.14: Normalized North Position Histograms and Gaussian Fit, $i = [304, 452]$.

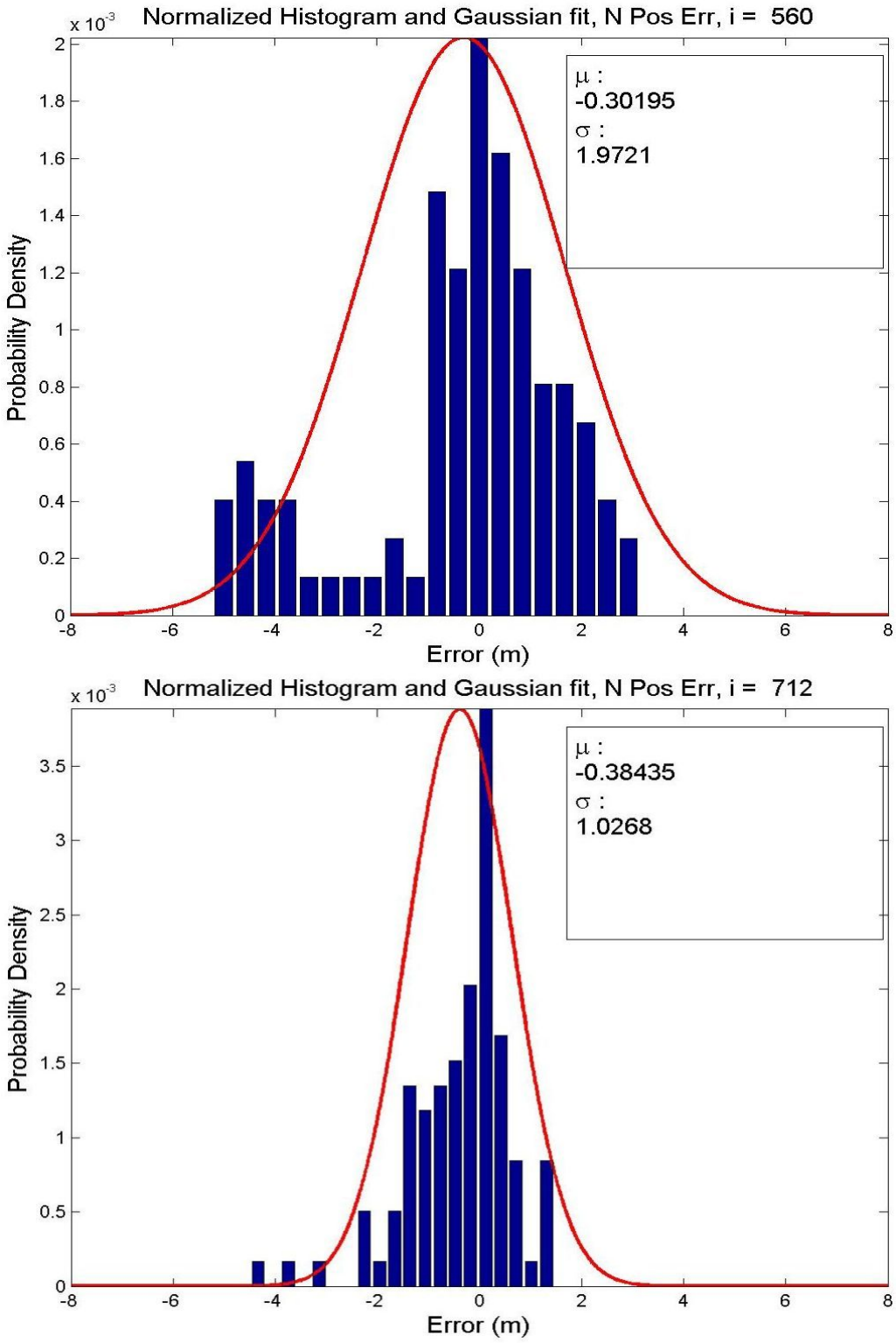

Figure 4.15: Normalized North Position Histograms and Gaussian Fit, $i = [560, 712]$.

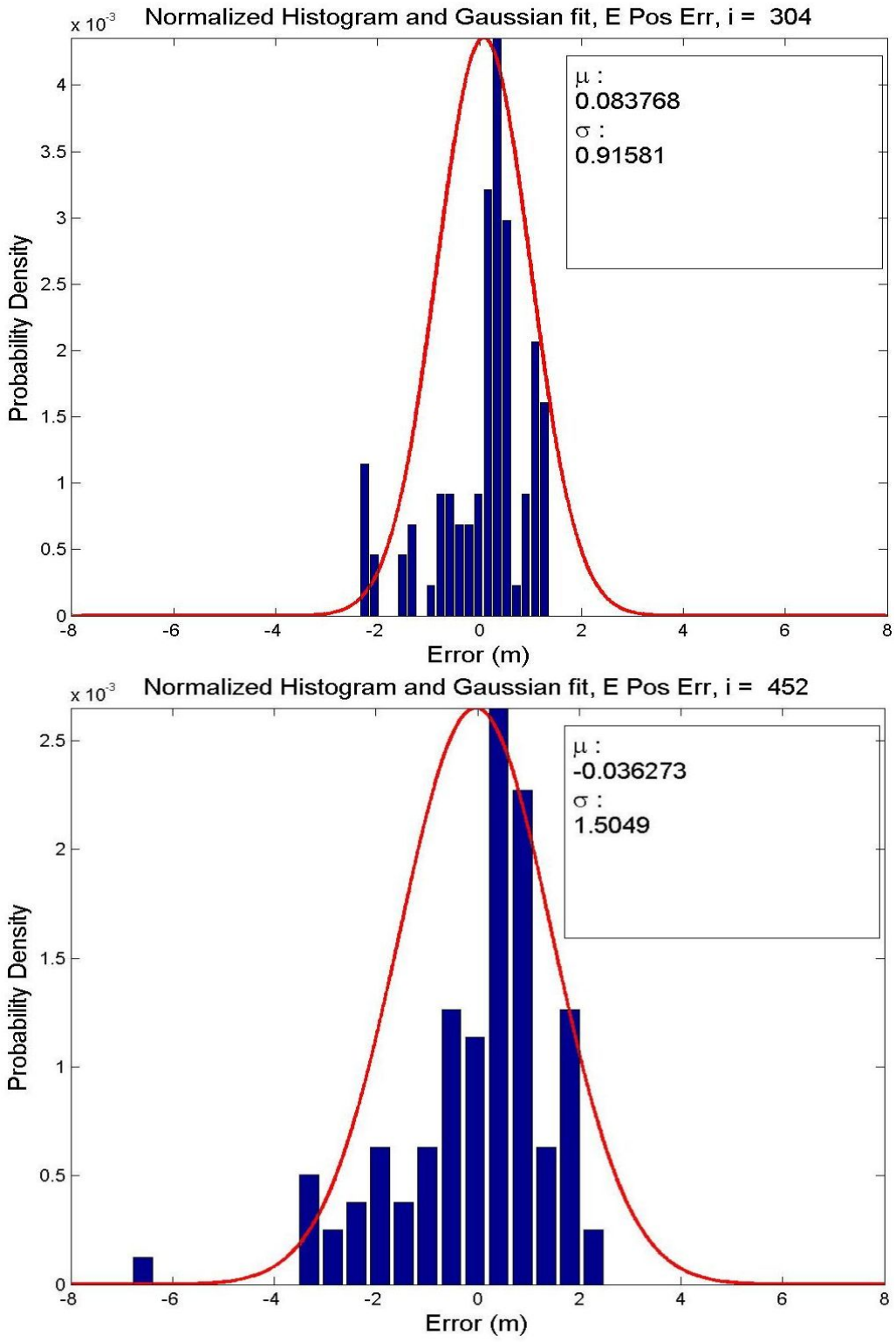

Figure 4.16: Normalized East Position Histograms and Gaussian Fit, $i = [304, 452]$.

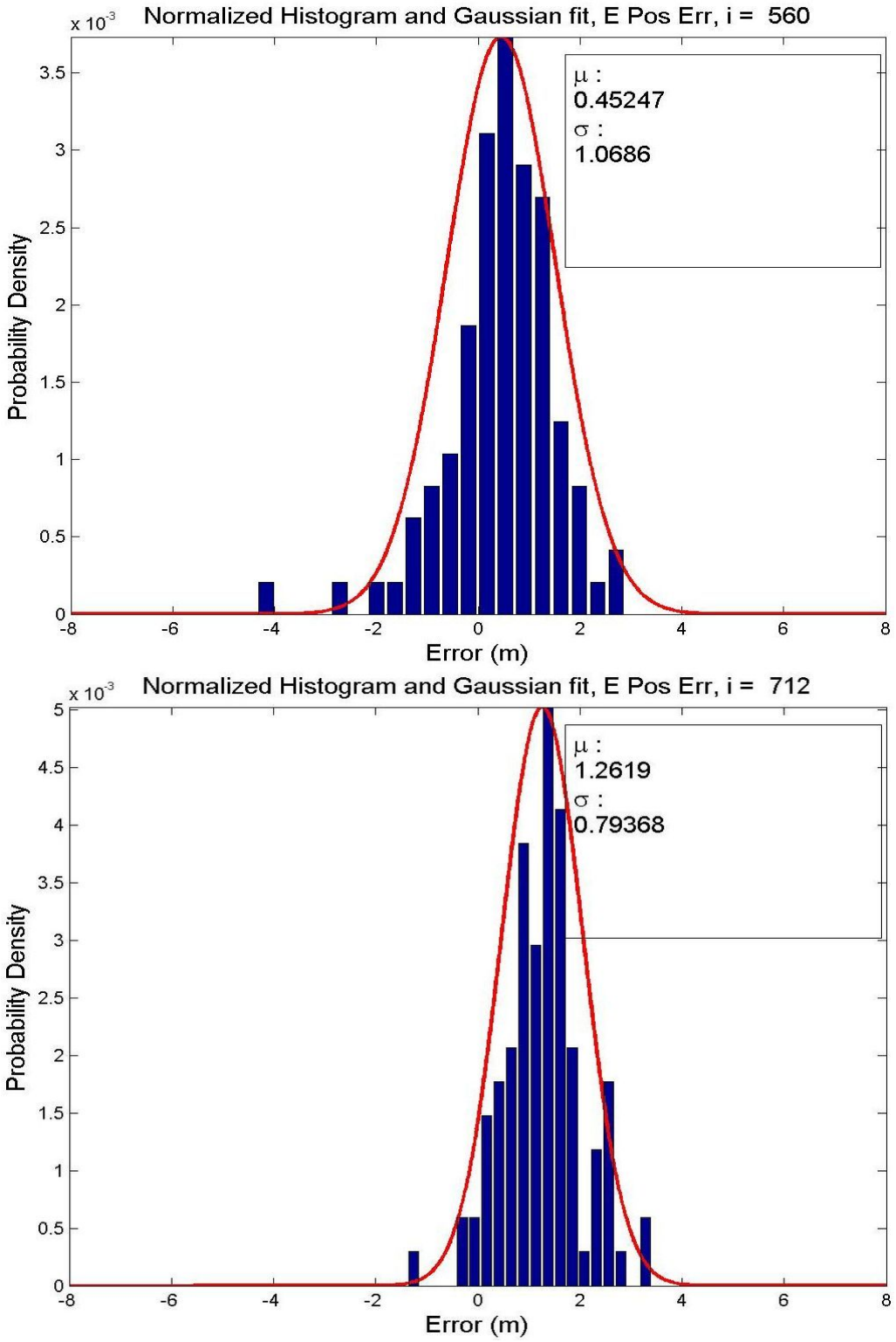

Figure 4.17: Normalized East Position Histograms and Gaussian Fit, $i = [560, 712]$.

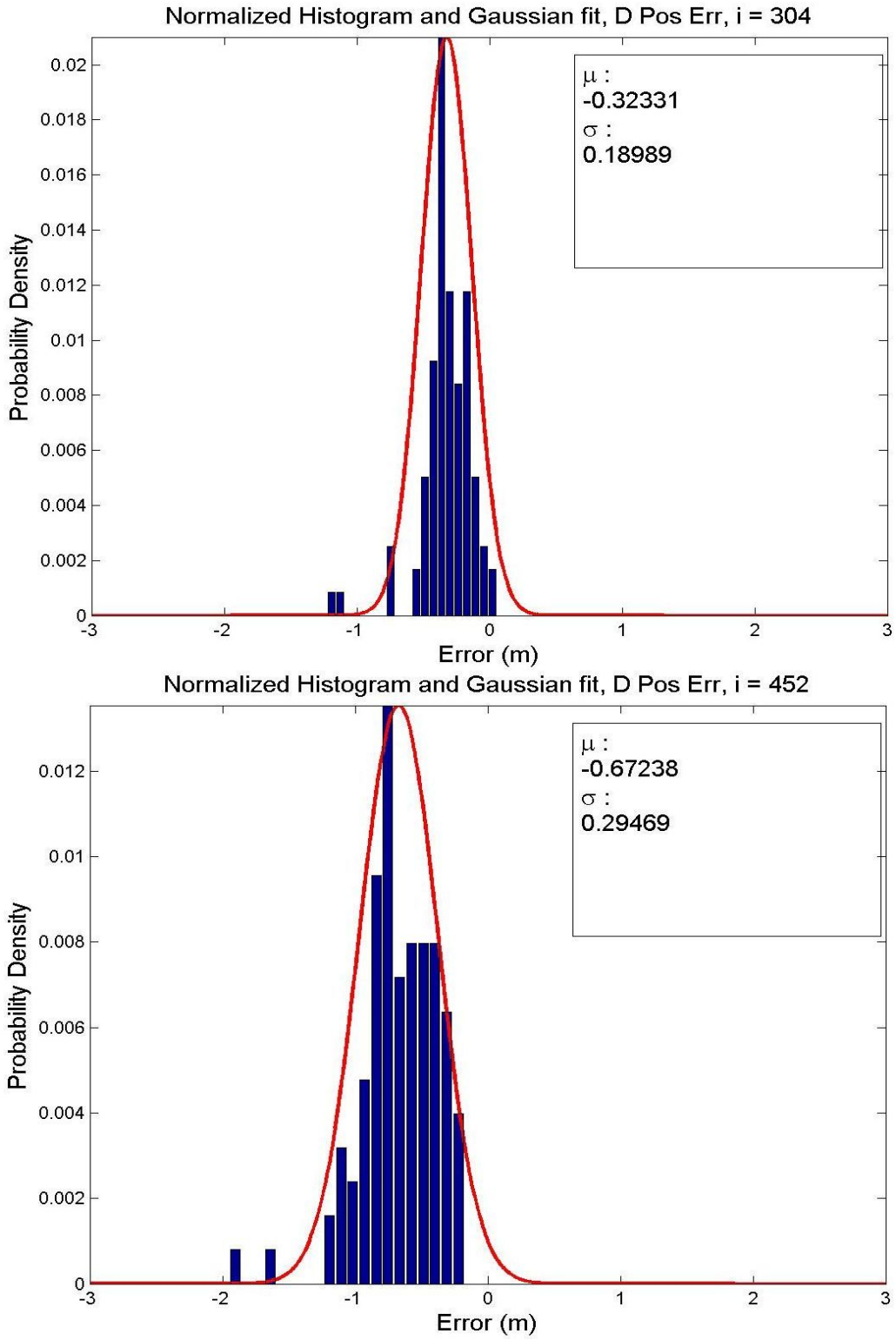

Figure 4.18: Normalized Down Position Histograms and Gaussian Fit, $i = [304, 452]$.

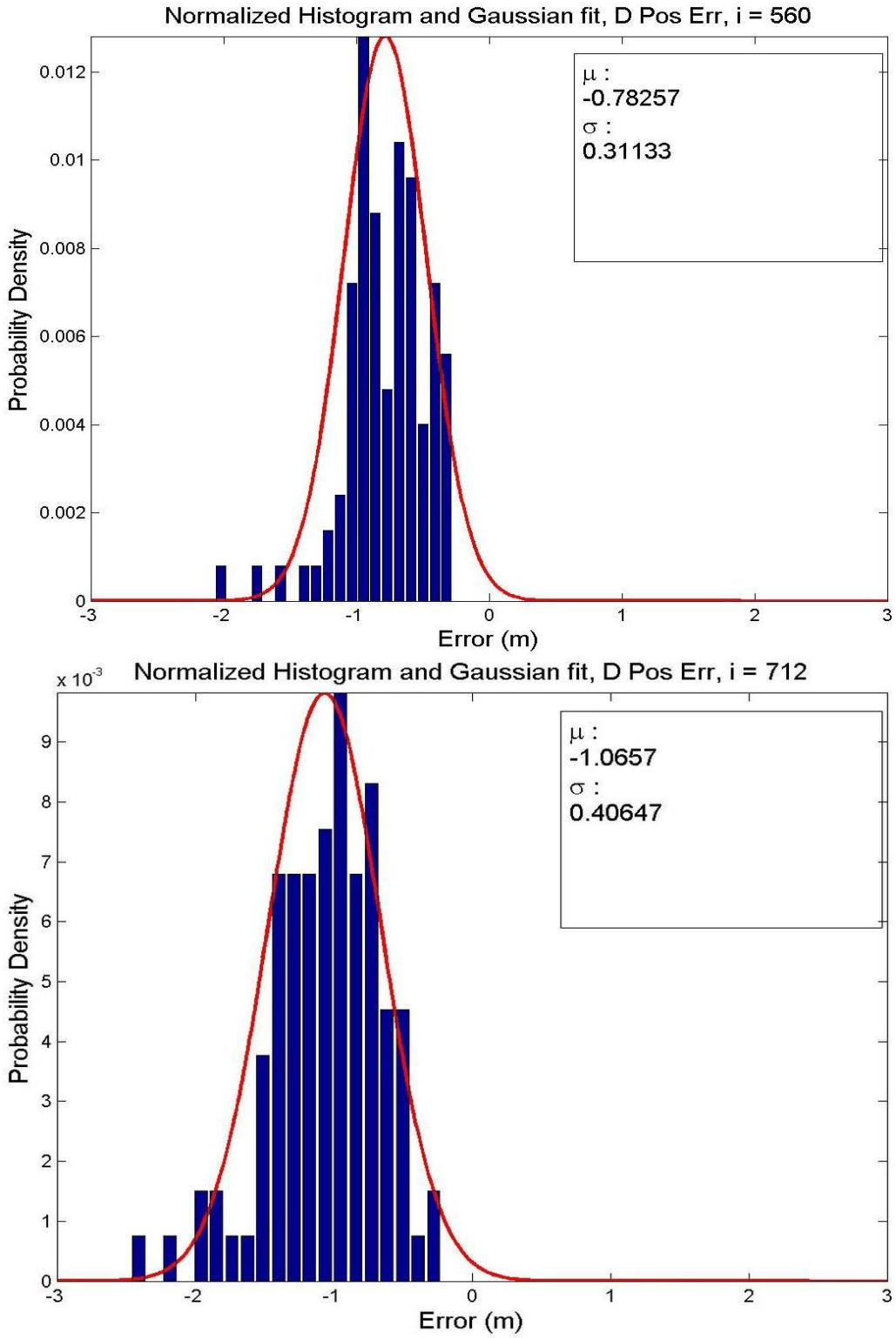

Figure 4.19: Normalized Down Position Histograms and Gaussian Fit, $i = [560, 712]$.

This same behavior is seen in the East position errors in Figures 4.16 and 4.17, although less pronounced. In all 4 samples, the core Gaussian shape is seen, with outlier errors skewing in the negative direction in the first three samples, before becoming more clustered at $i = 712$.

As for the downward error histograms and Gaussian fits shown in Figures 4.18 and 4.19, a Gaussian distribution of the errors does not hold, nor the Gaussian plus uniform. This is unsurprising given the disposition of the errors seen in Figure 4.8; it appears as if the distribution is nearly uniform over a given range, with approximately 8-10 outliers showing larger magnitude errors. This would suggest a better fit would be a distribution with a relatively flat crest and heavier tails.

4.3 Attitude Estimation Errors

In this section, the attitude estimation errors across all non-divergent runs are considered in a manner similar to that done for the positioning errors in Section 4.2. The estimation errors for roll, pitch and yaw are shown in Figures 4.20, 4.21, and 4.22, respectively. Again, the thick red line present in each of the plots is the mean error value.

The roll estimation errors shown in Figure 4.20 are seen to demonstrate consistent trends in the magnitude and direction of the errors. As the collection environment was indoors, the truth was based on the assumptions that the floor was perfectly level and the platform would not tilt or rock while taking a turn or coming to a stop. These assumptions translate to no change in roll or pitch for the duration of each run. Given the consistent movement in the errors over approximately the first 75% of each run, the errors seen in the plot may be due to the difference in the assumed flat floor and the slightly more bumpy floor that exists. Relative peaks at $i = [308, 456, 564]$ coincide with each left hand turn taken. The consistency of the errors becomes less evident as time progresses, with the

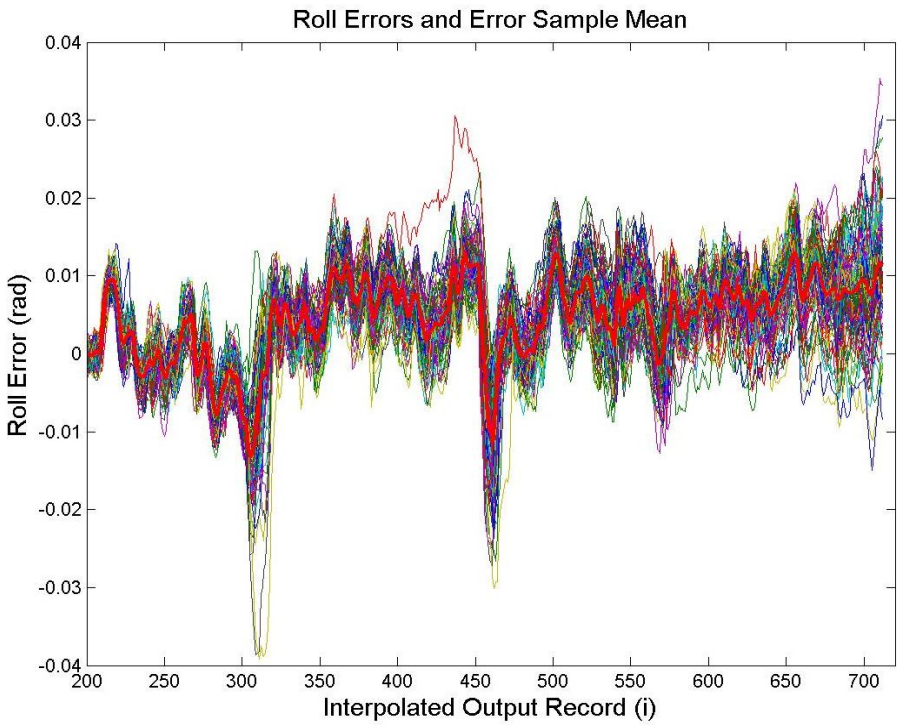

Figure 4.20: Roll Ensemble Errors.

errors becoming more noise-like, particularly after the third turn at $i = 564$. The errors are not zero-mean, but nearly so: the time average of the sample mean is 3.25 mrad.

The pitch estimation errors shown in Figure 4.21 are similar to those seen for the roll estimates, albeit noisier. The same flat floor assumptions that may be the primary drivers of the roll errors may also be the cause of the pitch errors as well. Additionally, it appears that 3 runs (seen near the bottom of the plot, in light and dark green) picked up a measurement bias coming out of the first turn. Overall, the time average of the estimation error is on the same scale as the roll errors, at -3.47 mrad.

The yaw errors are shown in Figure 4.22. It is seen that a large bias is present in the errors, with the mean error remaining fairly steady, with the exception of the times associated with the navigation of a turn. Around this mean, the errors appear noisy and Gaussian distributed. The large error spikes seen in the vicinity of the turns is due to the

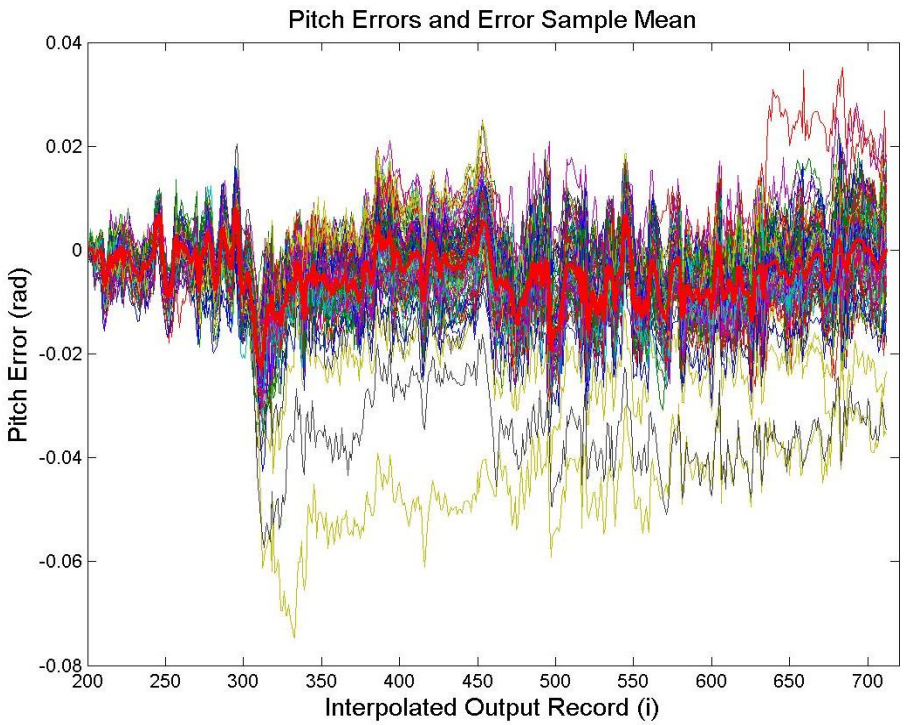

Figure 4.21: Pitch Ensemble Errors.

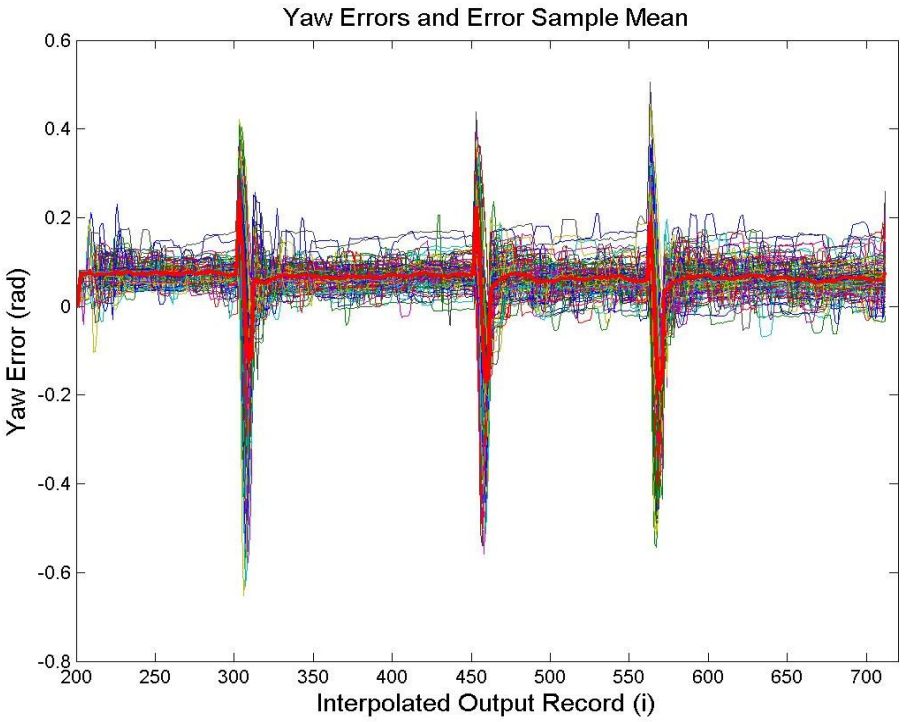

Figure 4.22: Yaw Ensemble Errors.

assumption of a constant turn rate through the turns, when in fact there is an acceleration and deceleration occurring. The most striking feature of the yaw errors, however, is their size relative to those seen in the roll and pitch estimates, being nearly a full order of magnitude greater.

Next, the sample mean $\pm 1\sigma$ plots are shown for the roll, pitch and yaw estimates in Figures 4.23, 4.24, and 4.25, respectively. The sample standard deviation for the roll and pitch errors are seen to increase with time until approximately the first turn, after which they remain relatively constant, rising slightly with each turn; the roll error sample standard deviation hits a peak of 8.2 mrad at $i = 311$, and a final value of 7.2 mrad. The pitch sample standard deviation peaks at a value of 12.1 mrad at $i = 496$, ending at 10.1 mrad. Neither decreases appreciably with time as did the North and East position sample standard deviations. The yaw sample standard deviation exhibits a similar behavior, growing slowly with time; however, the magnitude is again greater than that seen in either the roll or pitch, peaking at 245 mrad $i = 307$. The final value at $i = 712$ is 39.8 mrad, or about 2.3 degrees, more than 5 times that seen for the roll and nearly 4 times as large as the final pitch standard deviation value.

4.4 Covariance Optimism

As detailed in Section 2.10.3, one of the persistent problems seen in SLAM-EKF algorithms is the chronically undervalued filter computed covariance. While the magnitude of the horizontal filter errors have been shown in Section 4.2.2 to be generally under 1 m, an inaccurate filter computed covariance would cause any person or system to place too much trust in this result. An appropriate uncertainty measurement is critical if the filter estimates are to be used in navigation, either alone or in concert with other systems. This section considers the sample standard deviations for both the position and

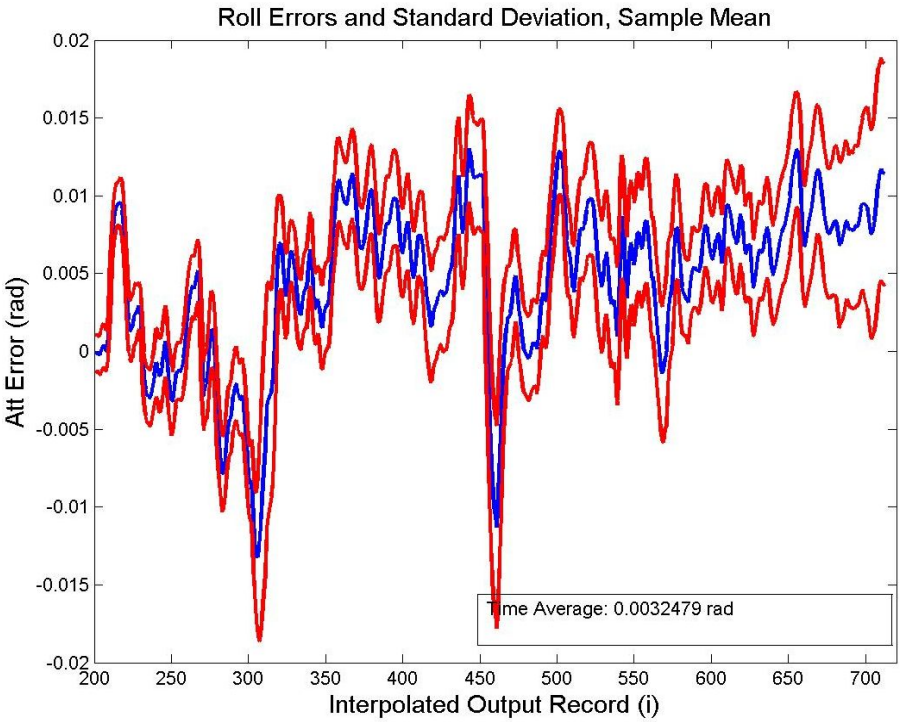

Figure 4.23: Roll Error Sample Mean and Standard Deviation.

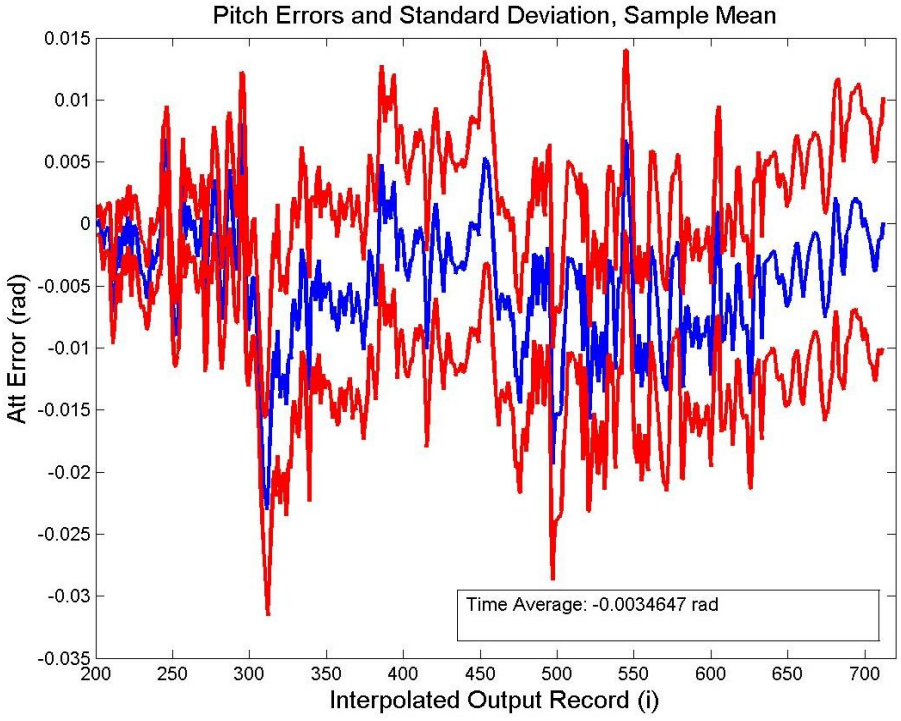

Figure 4.24: Pitch Error Sample Mean and Standard Deviation.

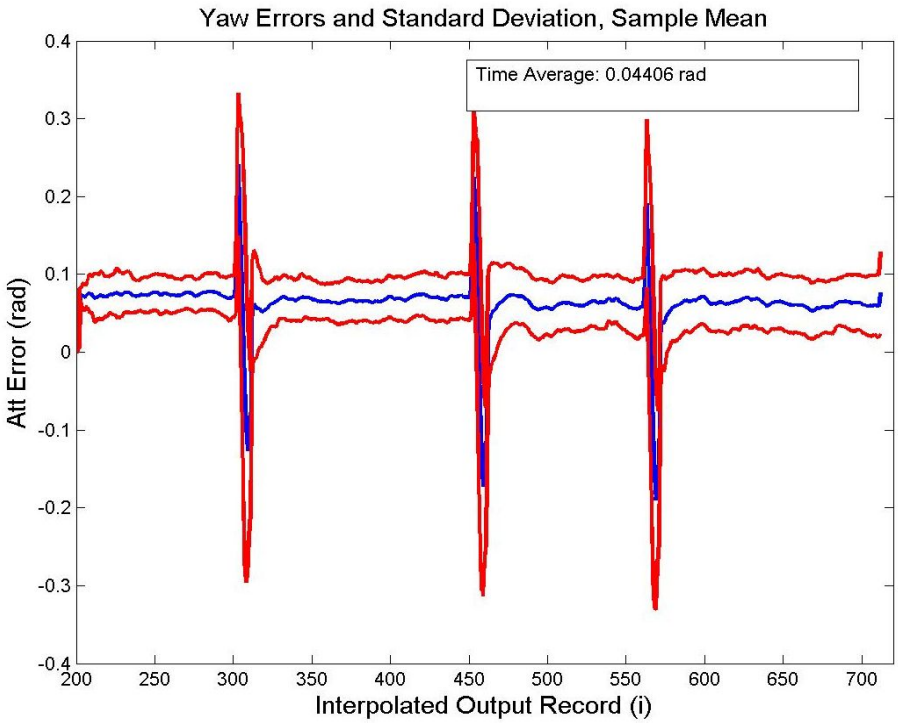

Figure 4.25: Yaw Error Sample Mean and Standard Deviation.

attitude measurements, calculated and presented in Sections 4.2 and 4.3, and compares these results with the filter computed standard deviations for the non-divergent runs.

4.4.1 Position Standard Deviations. This comparison begins with the illustration of the North, East and Down standard deviations, shown in Figures 4.26, 4.27, and 4.28, respectively.

The thick red line in each graph is the appropriate sample standard deviation of the errors, and the multitude of colored lines are the filter computed standard deviations, each line representing a different run. As is immediately clear from inspection of these plots, the general findings presented in 2.10.3 are verified; the filter vastly underestimates the uncertainty of the position estimates, with very few exceptions. Spikes in the filter computed uncertainty are seen at points corresponding to each of the turns. An increase in uncertainty is expected when new measurements are unavailable to the filter; this is often

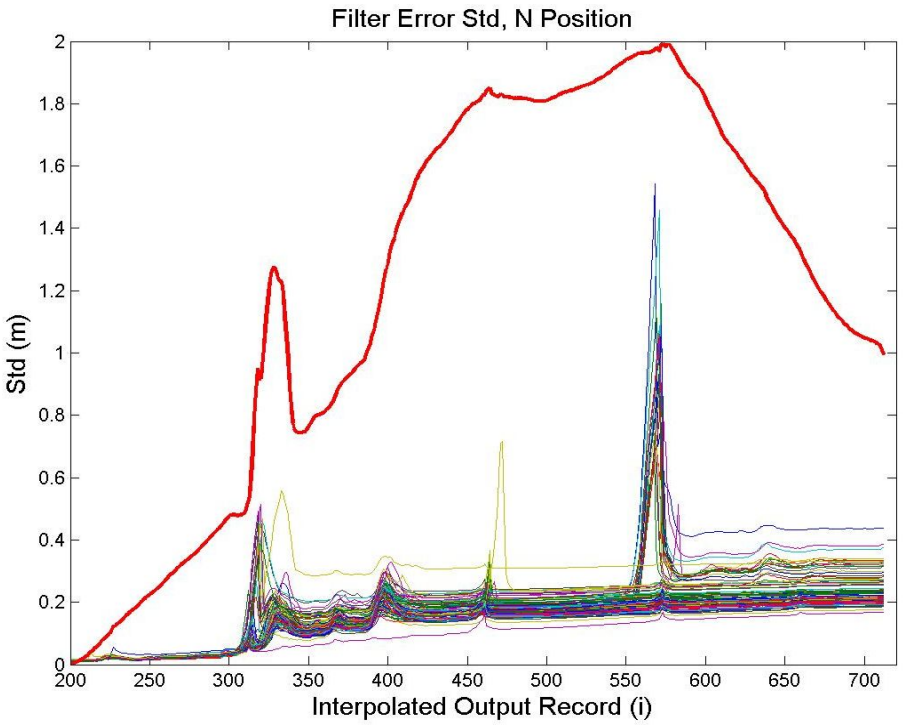

Figure 4.26: North Position Sample and Filter Computed Standard Deviations.

the case during the navigation of turns, as features are not within the camera frame long enough to be successfully tracked. This is especially the case in turn 3, where the walls are exceptionally bare. However, there is variation from run to run, and very large uncertainty spikes are restricted to a small subset of the runs under consideration. It is only in the cases of the East and Down positions that the surpassing of the sample standard deviation can be seen, and even then it is only seen in a small number of runs for a brief amount of time.

One metric available for use in the comparison of the sample standard deviation with that computed by the filter is the percentage difference between the two, denoted as D. The percentage difference may be calculated by taking the average across runs of all of the filter computed standard deviations and dividing by the sample standard deviation,

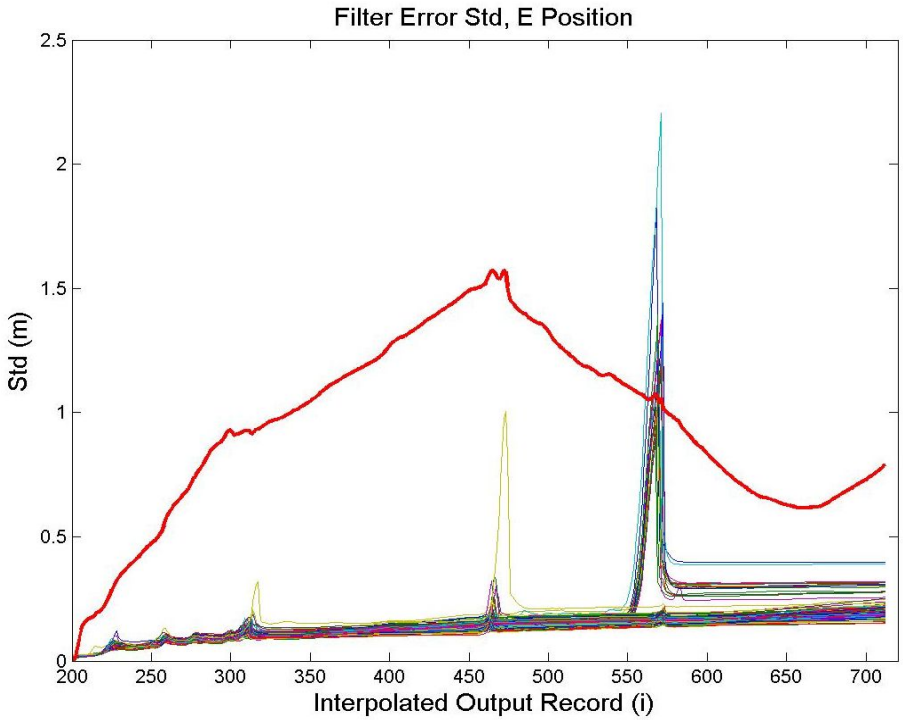

Figure 4.27: East Position Sample and Filter Computed Standard Deviations.

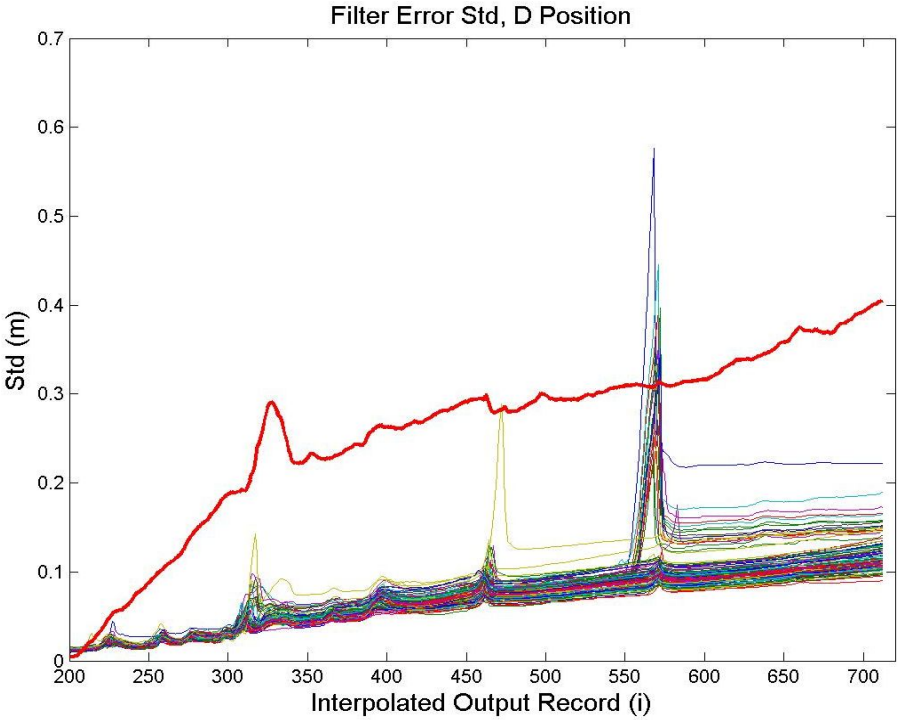

Figure 4.28: Down Position Sample and Filter Computed Standard Deviations.

$$D_i = \frac{100\%}{n * \sigma_{i,\text{sample}}} \sum_{k=1}^{n} \sigma_{k,i,\text{filter}} \tag{4.5}$$

where k denotes the run number, n the total number of runs, and i the sample time index. The results of this calculation for the position estimates are shown in Figure 4.29. Except for brief periods immediately following the alignment, it is apparent that on average, the filter is extremely optimistic with regards to the position standard deviation values. Taking the time average of the percentage difference values, it is found that the filter undervalues the North position standard deviation by 84%, the East by 79.9% and the Down position by 69.4%.

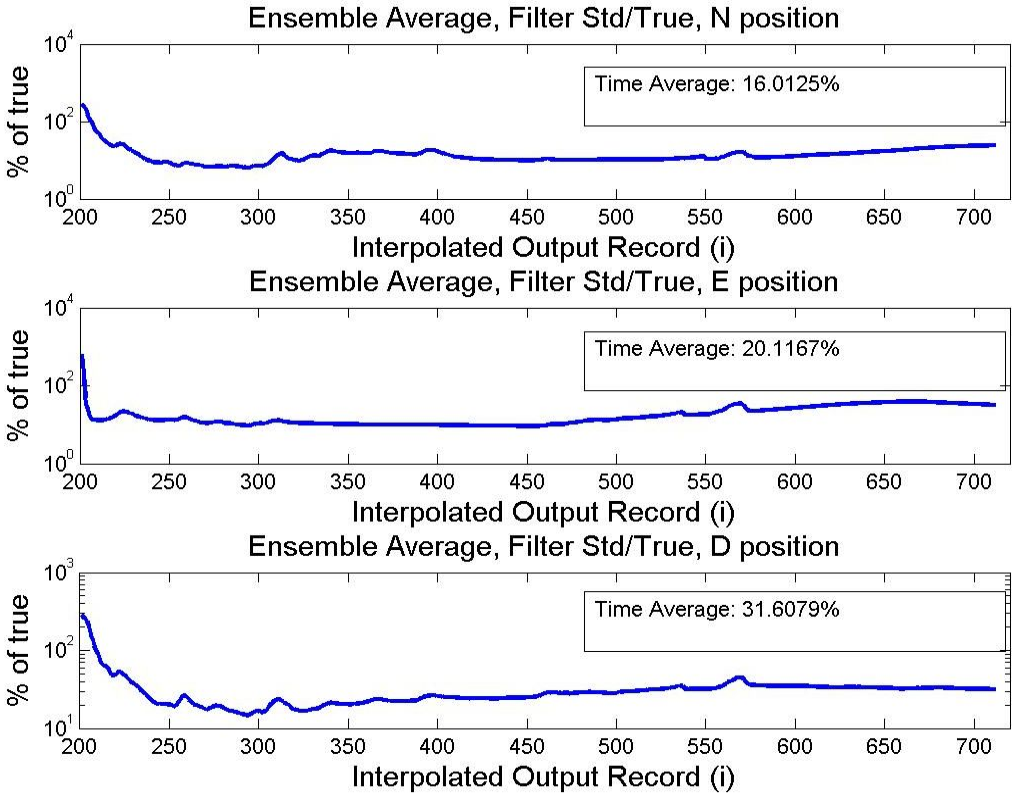

Figure 4.29: Percentage Difference Between Filter Computed and Sample Standard Deviation, Position.

4.4.2 Attitude Standard Deviations. As was done for the position estimates in Section 4.4.1, the filter computed standard deviations are shown with the sample standard deviations for roll, pitch and yaw in Figures 4.30, 4.31, and 4.32, respectively.

It is seen that the standard deviation estimate for the roll angle is relatively consistent before $i = 700$; the general magnitude and direction of the filter estimates tracks that of the sample standard deviation values. It is around the turns where the largest differences are seen. The filter computed values spike slightly, but not nearly enough in magnitude to match the sample values. Overall, this behavior represents a well-behaved EKF.

In Figure 4.31, the pitch exhibits a different behavior than that seen for the roll in Figure 4.30. From the first turn to the second, and from the third turn to the end of the path, there appears to be a multiplicative difference between the filter computed values and the sample of approximately 4; after navigating the second turn at $i = 456$, the filter

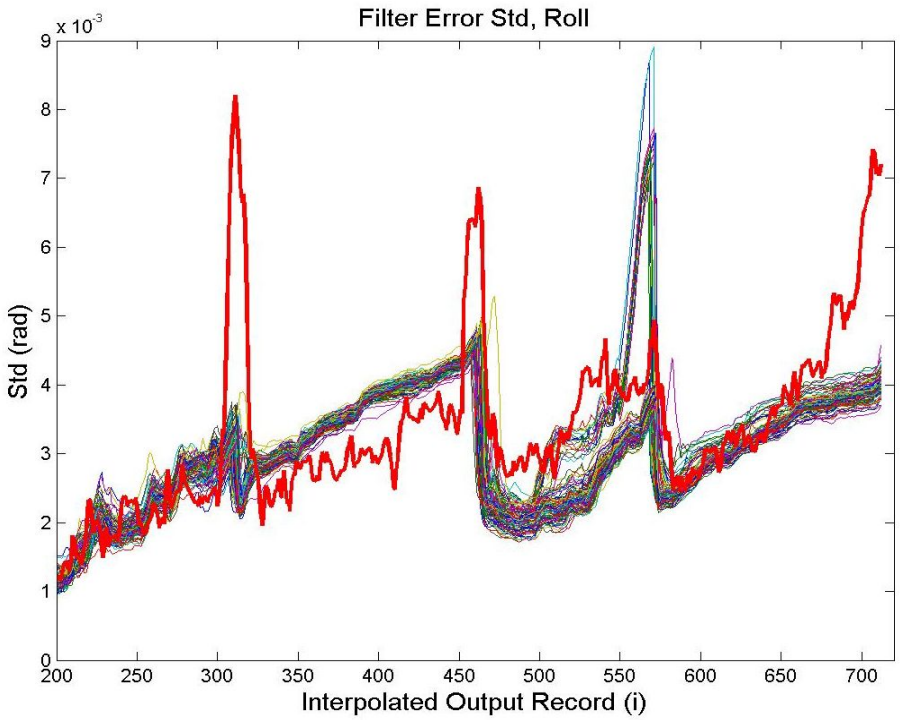

Figure 4.30: Roll Sample and Filter Computed Standard Deviations.

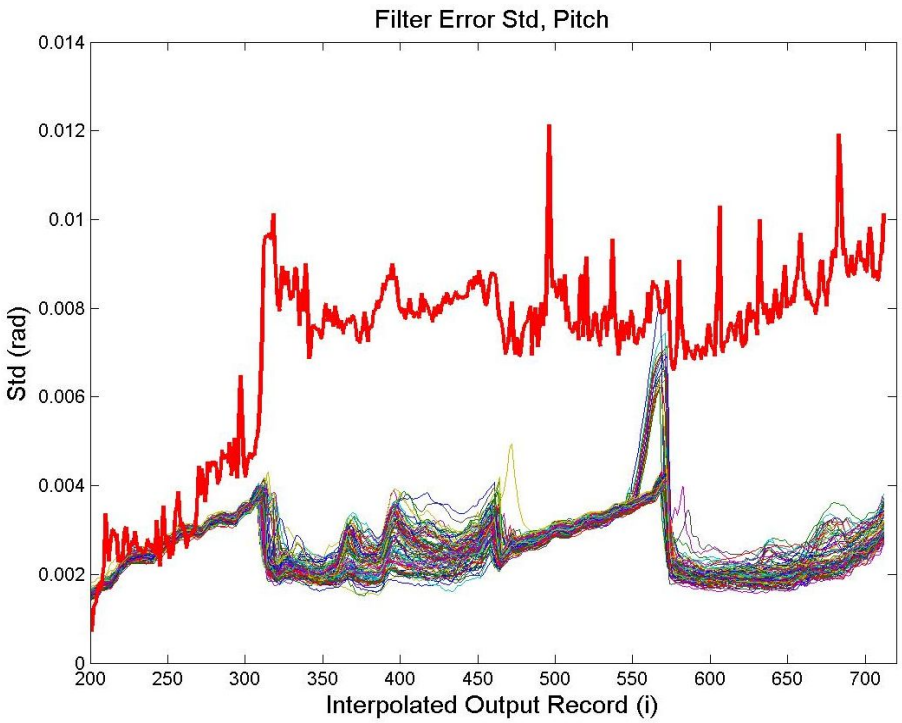

Figure 4.31: Pitch Sample and Filter Computed Standard Deviations.

less consistently follows the general shape of the sample standard deviation. In fact, near $i = 490$ there is a sizable spike in the sample value of which no trace is seen in the filter computed values. In a manner similar to that seen in the position and roll plots, a large spike in the filter computed standard deviation in seen in some runs in the area surrounding turn 3.

The larger magnitude of the sample standard deviation seen through the turns in Figure 4.32 exaggerates the appearance of the filter inconsistency. These large spikes are due to the larger magnitude errors that were seen in the turns due to the constant turn rate assumption in the truth, and are not an accurate reflection of the standard deviation ought to be at these points in time, though in general an increase of uncertainty in these time periods would be expected. A more realistic view is represented by the sample standard deviations seen over the straight path segments. As in the position and the pitch estimates, the filter computed standard deviation is for all runs far below the sample value.

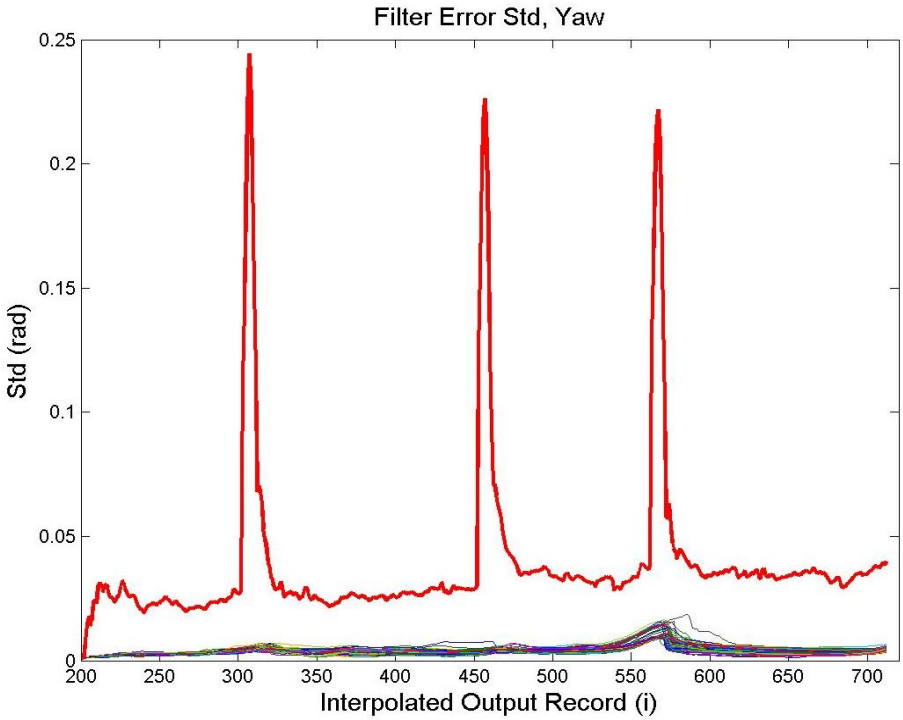

Figure 4.32: Yaw Sample and Filter Computed Standard Deviations.

In Figure 4.33, the percentage difference between the filter calculated and sample standard deviations, obtained through application of Equation 4.5, is shown. As was stated above in the consideration of Figure 4.30, the filter appears to estimate the roll standard deviation quite consistently; the filter underestimates the standard deviation by only 4.5%, when a time average of the percentage differences is taken. For the pitch angle, the filter is off by 56.3%, with the filter doing a much better job before a turn is taken as compared with the rest of the run. The filter is off by 87.7% for the yaw. Even discounting the time periods where the large magnitude errors are present, the filter never estimates above 24.1% of the sample value.

4.4.3 ANEES. Consideration of Figures 4.26 through 4.33 makes it immediately clear that the filter is optimistic. A measure of the degree of optimism is found through the

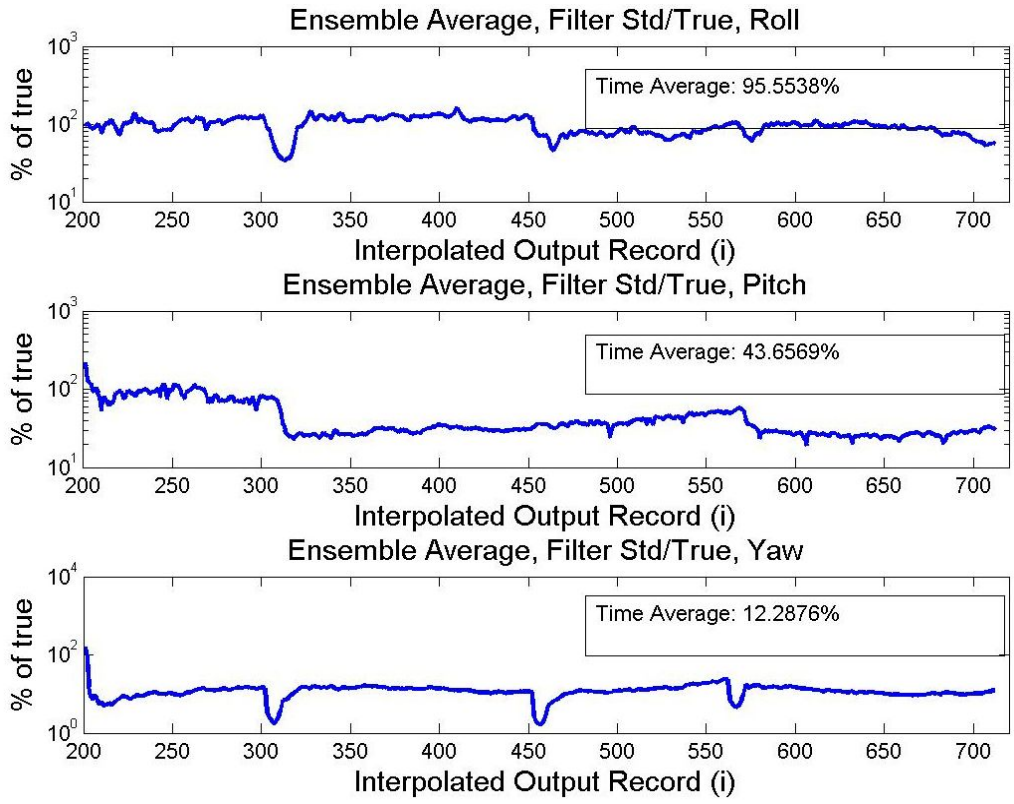

Figure 4.33: Percentage Difference Between Filter Computed and Sample Standard Deviation, RPY.

calculation of the ANEES through application of Equations 2.66 and 2.108. If the filter is linear-Gaussian and consistent, then $N\eta(t_i)_{\text{avg}}$ will lie within the bounds of a χ^2 distribution with an appropriately selected confidence interval. The degrees of freedom is equal to N times the length of the state vector, as discussed in [1] and [2]. Given 98 runs, a state vector of length 3 (position only), and a confidence interval of 0.95, the χ^2 bounds are determined to be approximately $[2.35, 3.74]$. The appropriate bounds and ANEES statistics for the 98 run set are shown in Figure 4.34.

The appearance of Figure 4.34 verifies the fact that the filter is extremely optimistic. The position only ANEES, shown as the blue line in Figure 4.34, first spikes just after the alignment period then slowly begins to 'settle' near a value of 250, over 60 times the value of the upper bound. The filter does not become inconsistent over time as do those

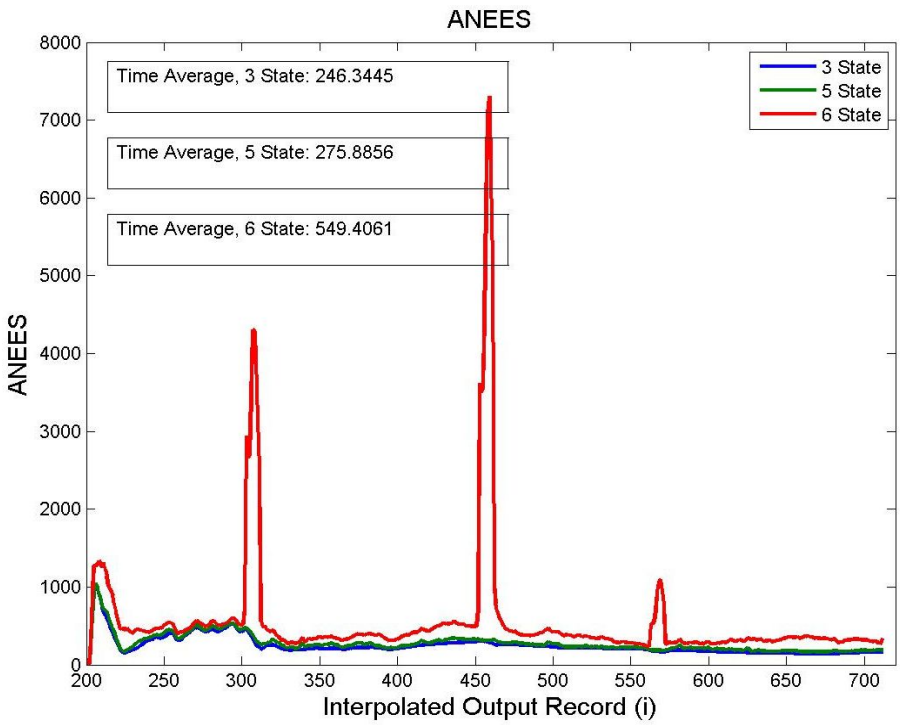

Figure 4.34: ANEES over 98 runs. The blue line represents the ANEES of the position states only, the green is the position plus roll and pitch states, and the red includes all position and attitude states. The cyan lines at the bottom of the plot are the χ^2 distribution bounds for the three state case. Due to the scale of the ANEES values, these bounds are indistinguishable from the x-axis; the filter is clearly inconsistent.

considered in [1], but are immediately and grossly optimistic.

In contrast with [13], this work considers not only the position estimates and associated characteristics, but the attitude as well. In support of this, the ANEES statistic is again calculated including the attitude states. The results of these calculations are shown in Figure 4.34. The green line is the ANEES including the position states as well as the roll and pitch, while the red line incorporates the yaw estimates as well. As might be expected based on the appearance of Figures 4.32 and 4.34, inclusion of the attitude estimation errors and covariance results in a substantial increase in the ANEES calculation, primarily during the navigation of the turns at $i = [308, 456, 564]$. The time averaged and peak values of the 6-state ANEES are 549.4 and 7309.8 respectively,

compared to 246.3 and 1022.7 for the position only ANEES. Inclusion of the attitude states more than doubles the time averaged ANEES; however, the yaw estimate is the primary driver behind this increase. If the yaw estimates are removed and a 5-state ANEES calculated, the time averaged and peak values are 275.9 and 1050.0 respectively, relatively close to the position only ANEES values. This confirms the assertion that the yaw estimates are the overall largest contributors to filter inconsistency. While this level of inconsistency is certainly undesirable, some solace may be found in the fact that for the given run length, the filter does not appear to diverge due to this issue, as discussed in the next section. It is expected that the inconsistency *would* eventually lead to divergence given a longer run length, based upon the findings of [1] and [7], but this is as yet unconfirmed for this IAN algorithm.

4.5 Divergent Runs

As shown in Figures 4.2 and 4.4 in Section 4.1, 2 of the 100 analyzed runs were shown to diverge in the position estimate as the platform was navigated around the second turn. These runs are examples of the classic 'something went wrong' result; the divergent behavior is unexpected and atypical, and the results dismissed. One result of this work is that a measure of the rate at which these divergent runs occur for this filter algorithm has been experimentally determined. To further explore the divergent runs, this section considers runs 40 and 75, separately; relying on the observed behavior, measurement residuals and covariance, the causes of divergence in each run are explored. The results of attempts to correct the filter behavior for each run are also presented.

Before beginning the examination of the causes of filter divergence in each run, it is suitable to consider whether or not there was any indication in the filter that divergence had occurred. The filter computed covariance is an appropriate place to look for such an indicator; as this is a measure of filter uncertainty, the covariance should increase as the estimate becomes poorer. In fact, the filter computed covariance and residuals are the only

metrics available in a real world situation that may indicate a problem with the filter solution. In Figures 4.35 and 4.36, the filter computed standard deviation σ is shown for all runs, similar to that shown in Figures 4.26 through 4.32, with the exception that the filter computed standard deviation from the divergent runs is included. The calculation of the sample standard deviation (the thick red line) also includes these values. It can be seen in that in all cases, the standard deviation in the divergent runs grows rapidly after the second turn at $i = 456$. It similarly drops rapidly as the third turn is navigated; in the position estimates, the standard deviation of the divergent runs appears to settle on a value after the third turn. In the attitude estimates, the standard deviation drops to levels similar to those seen in other runs; all indications that something has gone wrong essentially disappear. While a significant rise in the filter computed covariance occurs after divergence, a significant number of updates occur before the filter computed covariance 'catches up' and begins to significantly differ from the values computed for the non-divergent runs. In other words, an indication of divergence is contained within the filter computed covariance, but the associated increase in magnitude does not occur quickly enough to be used as a timely divergence indicator. Thus, a traditional covariance monitoring scheme would be an unsuitable divergence detector in these cases.

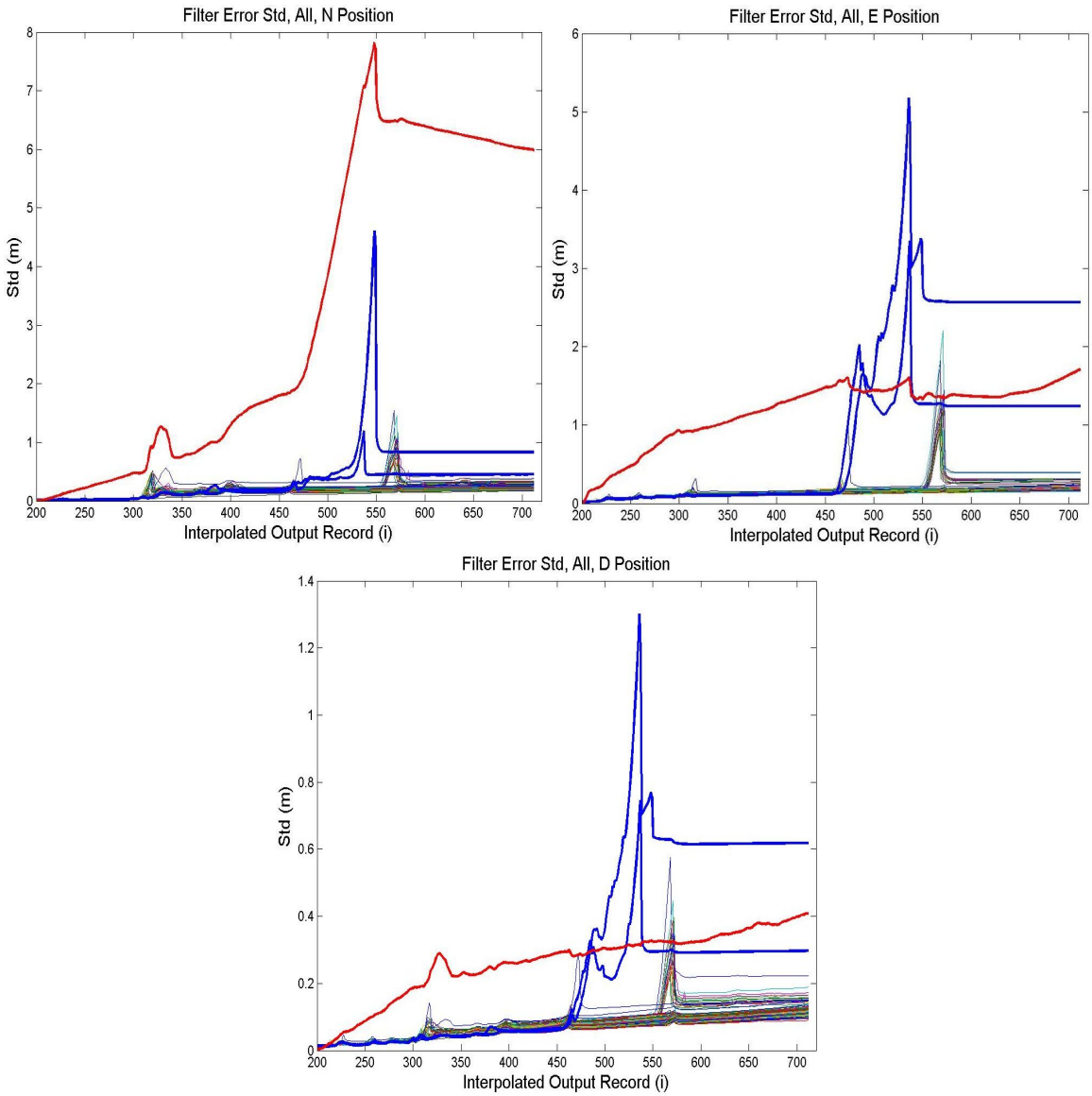

Figure 4.35: Position standard deviations, including divergent runs. The thick red line is the Monte Carlo standard deviation. The multi-colored lines near the bottom of each plot are the filter computed standard deviations for the non-divergent runs, and the two thick blue lines are the filter computed standard deviation for the divergent runs.

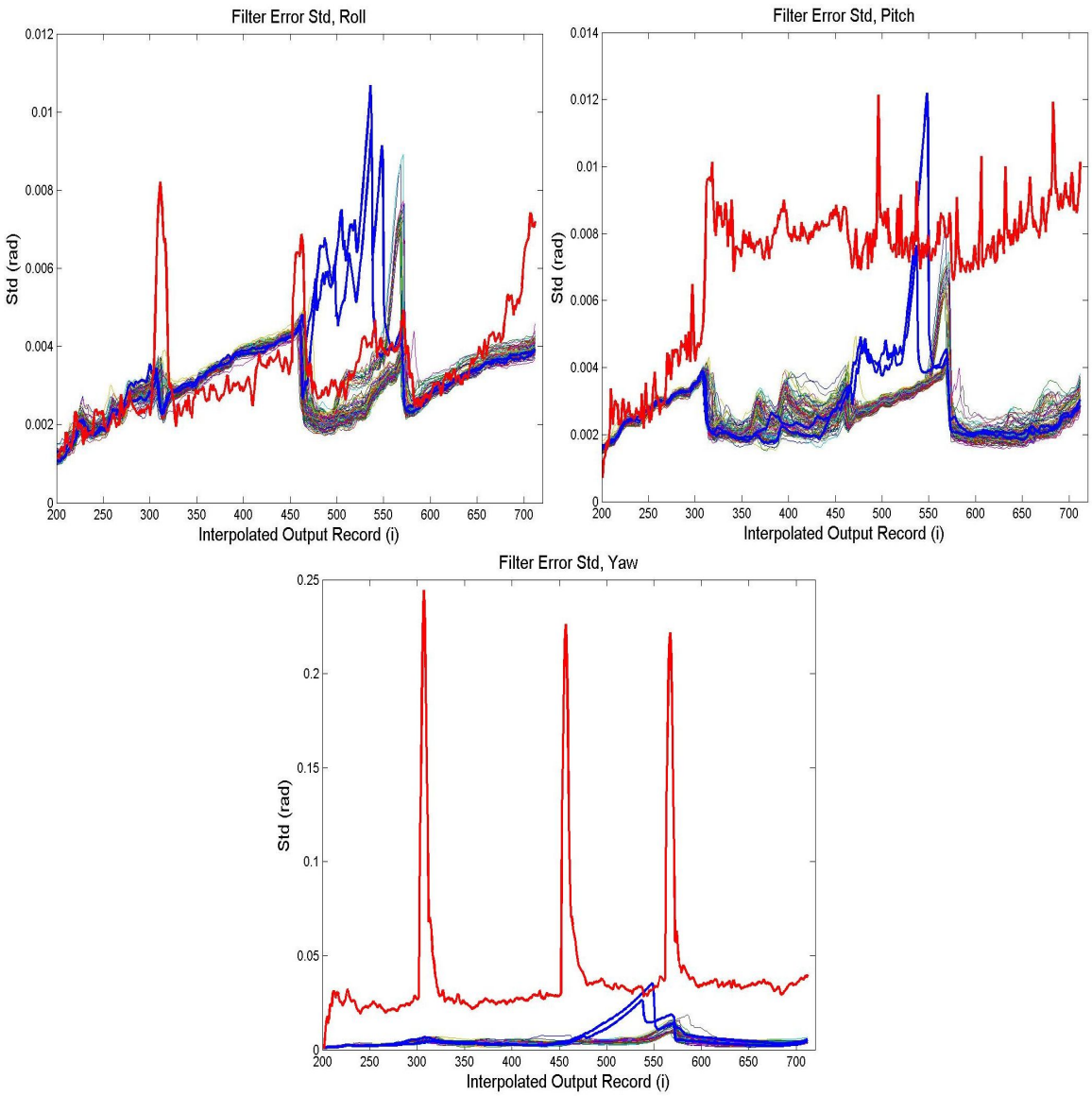

Figure 4.36: Attitude standard deviations, including divergent runs. The color scheme for the lines representing each metric (Monte Carlo, divergent, and non-divergent standard deviations) is the same as that described for Figure 4.35.

4.5.1 Run 40. In this section, run 40 is given special consideration. As seen in Figure 4.37, the filter position estimates begin to diverge approximately halfway through the filter run time. The platform approaches a corner and the scene is populated by a set of double doors, a large window to the right, signs on the ceiling and walls, and lighting fixtures. Features are tracked as normal, and then the platform navigates through the turn. This sequence of images can be seen in Figures 4.38 through 4.42. The magenta circles are predicted feature locations, and the yellow stars encircled in blue are features that are successfully matched to the associated predicted feature.

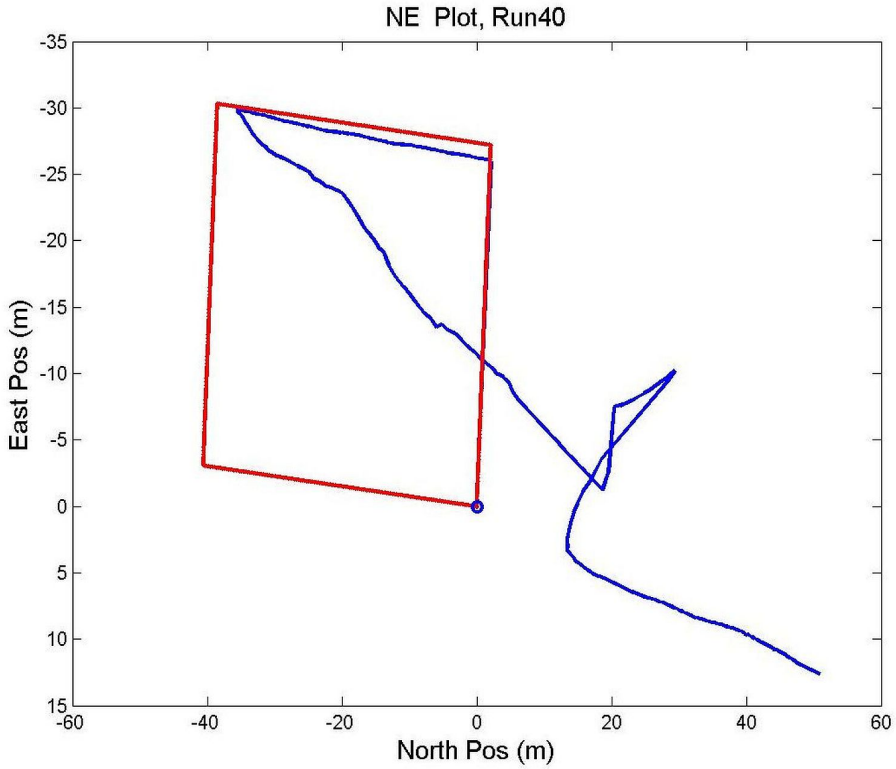

Figure 4.37: Run 40 N vs. E Plot, Diverged.

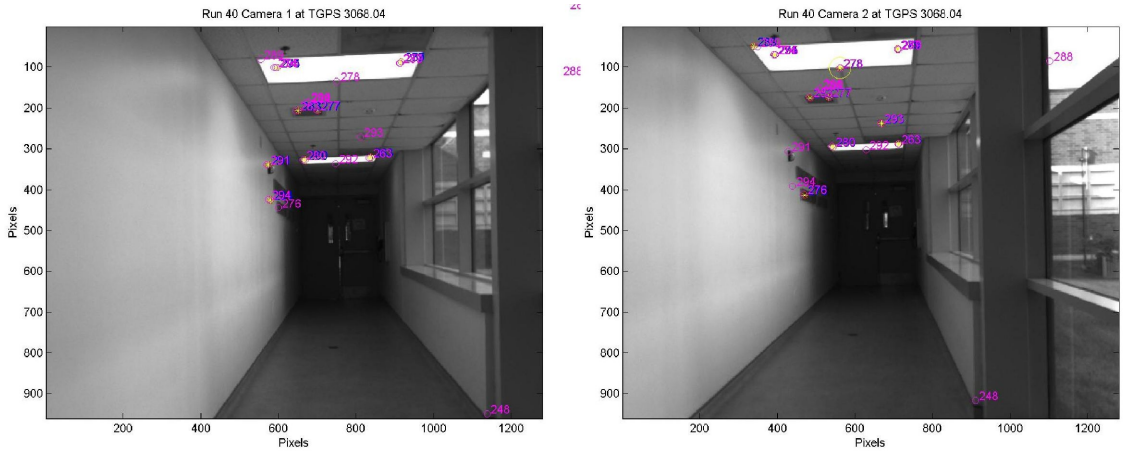

Figure 4.38: Run 40 Tracked Features, t = 3068.04 s.

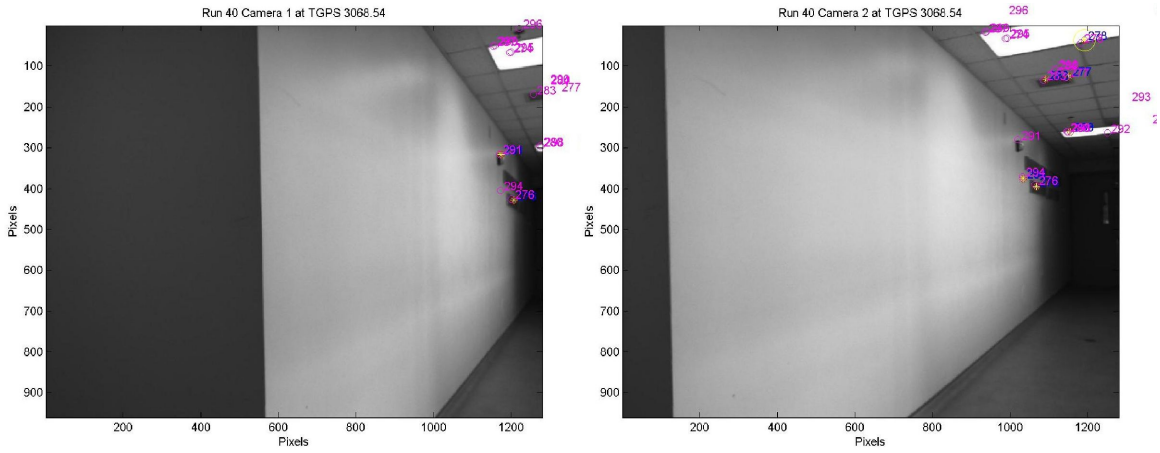

Figure 4.39: Run 40 Tracked Features, t = 3068.54 s.

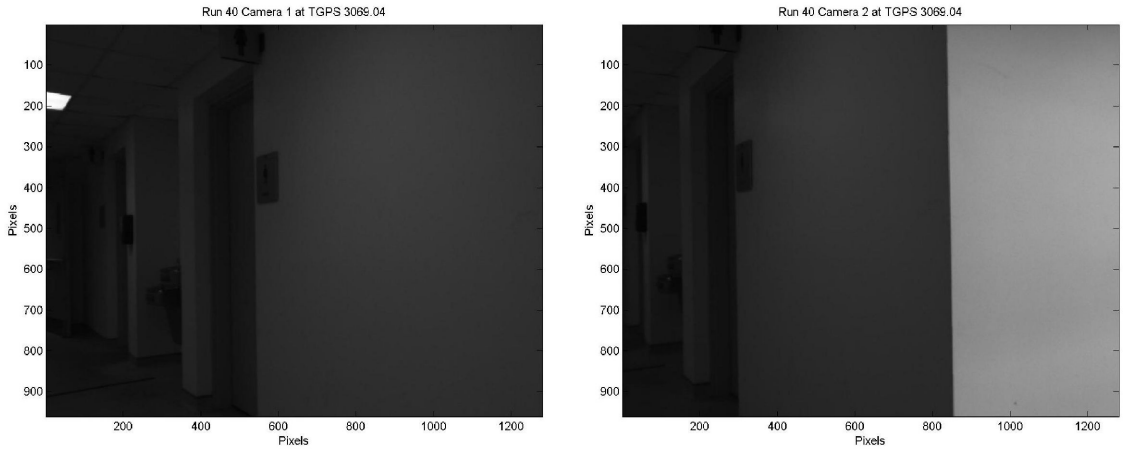

Figure 4.40: Run 40 Tracked Features, t = 3069.04 s.

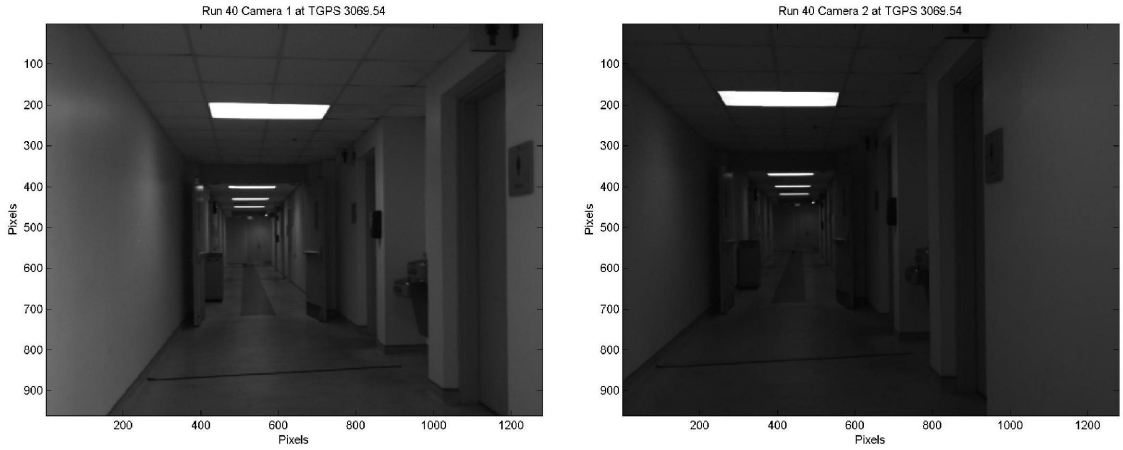

Figure 4.41: Run 40 Tracked Features, t = 3069.54 s.

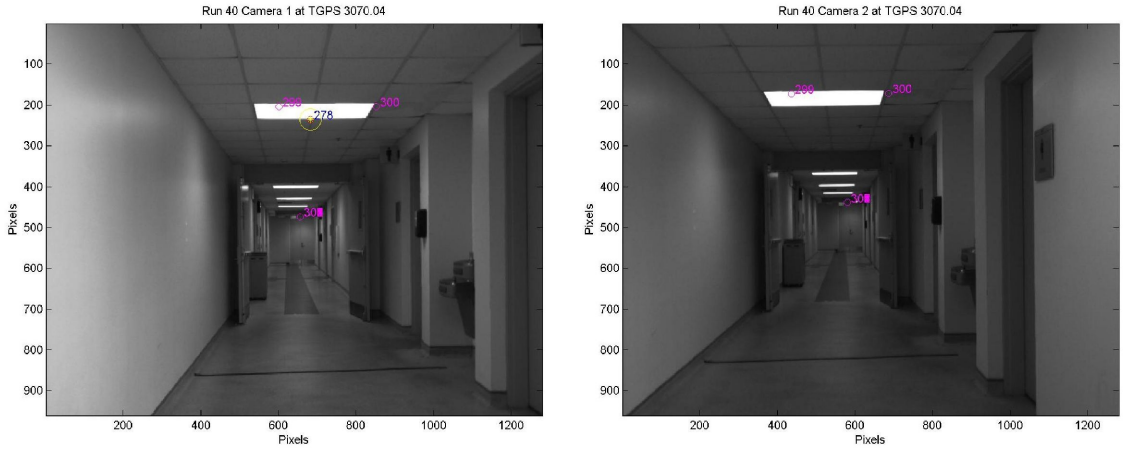

Figure 4.42: Run 40 Tracked Features, t = 3070.04 s.

There are essentially two information sources for the filter: the inertial measurements and the estimated pose obtained through feature prediction and matching. As it seems unlikely that the IMU is suffering from a catastrophic failure at the same physical location in two runs and no others, it is prudent to assume the cause of the divergence may be found in the image updates. Figure 4.43 shows the residual magnitude in pixels, for features matched in the time surrounding the second turn. Noticing the scale on the figure, it is seen that two updates have extremely large residual magnitudes, dwarfing all others.

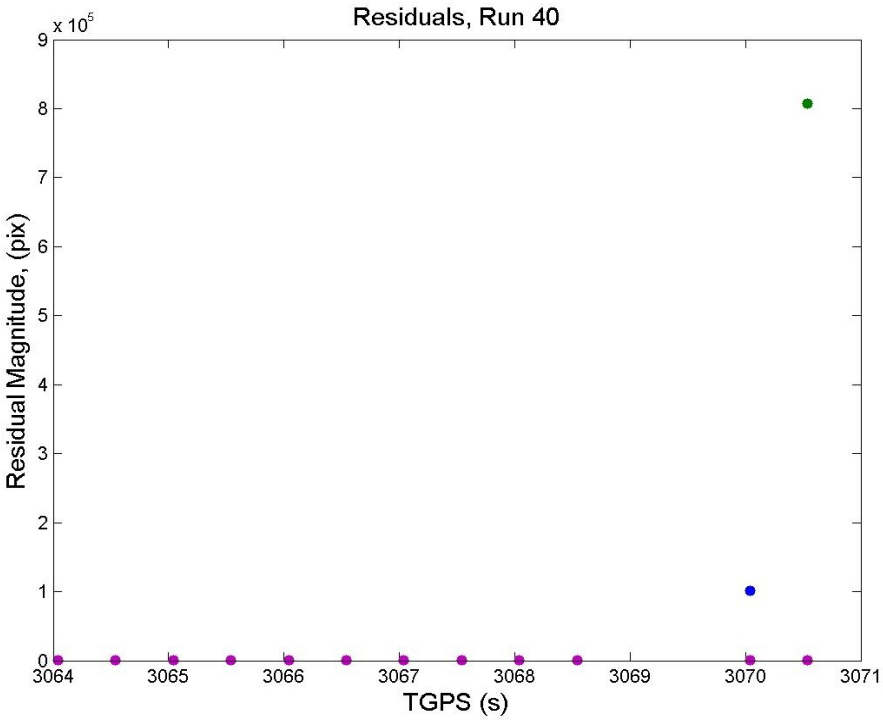

Figure 4.43: Run 40 Residuals.

The first of these occurs at $t = 3070.04$, seen in Figure 4.42. The feature corresponding to these residuals is 278, found on a lighting panel and surrounded by a yellow circle in Figures 4.38 and 4.39. After being successfully matched twice, there is a 1 second period during which no updates occur, and the filter uncertainty greatly increases, as expected. Finally, feature 278 is matched again; however, *it is matched to a point on an altogether different lighting panel*; see Figure 4.42.

Within this IAN algorithm, a successful feature match must satisfy 3 conditions as discussed in Section 2.9.3: first, the stochastically weighted distance between a predicted feature location and a possible match must fall below a given threshold; second, the feature descriptors of the potential match and tracked feature must be beat an arbitrary similarity threshold; and finally, the best potential match after the first two constraints have been met must be sufficiently 'unique' when compared to the next best potential match. The specific reason this match occurred is still under investigation; what has been

determined is that in run 40, the statistical weighting process resulted in all 90 potential features meeting the first condition with respect to tracked feature 278. The distance between the predicted feature location and the candidate features inhabited a range of [1.1264,1.2637] pixels, while the threshold was set at 2.15 pixels. One of these candidate features happened to be similar enough to the tracked feature and sufficiently distinct from the other candidates to allow a match. When comparing the distance between predicted feature locations and candidate matches, the weighting is reliant on both the uncertainty of the predicted feature location and the measurement uncertainty, both of which significantly increased following the period t = (3069.04, 3070.04) during which no updates occurred. It must be emphasized that this case of filter divergence is not caused by a general IAN problem, but is specific to this run and the method of statistical weighting used in this algorithm. Altering Equation 2.102 with a term that penalizes candidates based upon the magnitude of their unweighted residuals may have been sufficient to prevent the match that caused divergence in this case.

As noted in Section 2.10.1, the filtering algorithm includes the capability to drop 'stale' features that have not been matched for a specified period of time. During the course of this research, this length of time was kept at 1 second, and any feature that is not matched for *more* than one second is dropped from the state vector. In this case, feature 278 was not matched for exactly 1 second, and then matched in the following frame. Therefore, if the feature were to be kept from being matched incorrectly for another time step, it would be considered stale and removed. Further, if the hypothesis that this update caused the filter to diverge were correct, preventing this update would result in improved filter performance. One method would be to lower the length of time a feature is tracked from 1 second, to force the removal of the spurious update. This method carried the possibility of significantly altering the filter performance in the time prior to the tracking of feature 278, and was thus not acceptable. Rather, the image set corresponding to

t = 3070.04, the time of the first spurious update, was made unavailable to the filter. This forced feature 278 to be dropped and prevented the bad matches from occurring. Figure 4.44 shows the result when this course of action was taken. The IMU measurements were simply propagated over this period of time until another image set became available. It can be seen in Figure 4.44 that although large magnitude errors remain in the North direction, the filter no longer diverges. In this case, residual monitoring could have allowed for this update to be rejected, and divergence avoided.

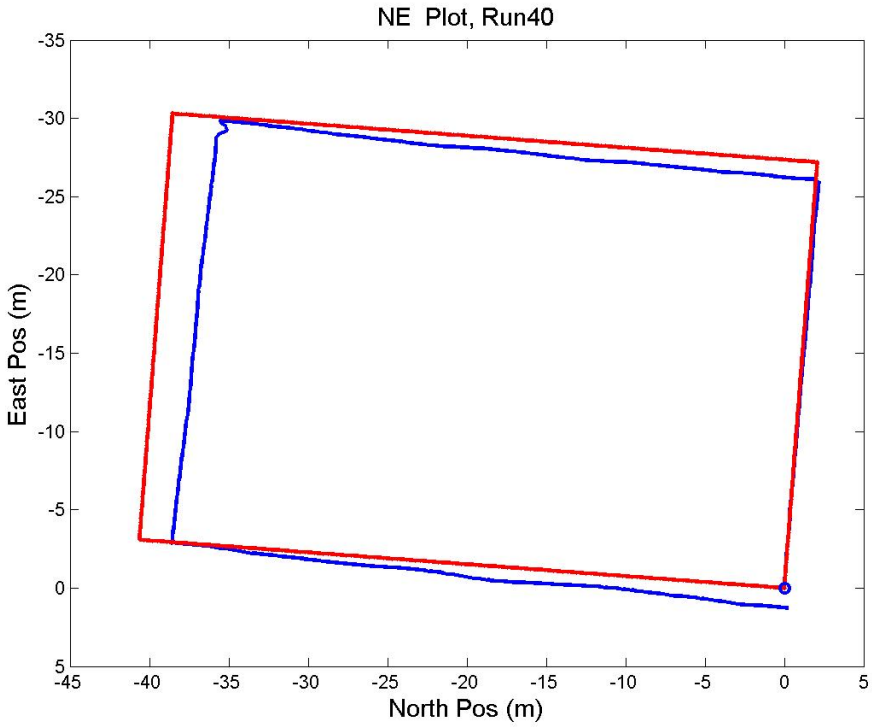

Figure 4.44: Run 40 N vs. E Plot, Images Removed.

4.5.2 Run 75. As done for run 40 in Section 4.5.1, run 75 is here analyzed for causes of divergence, and attempts to remedy the problem are addressed. The divergent NE plot for run 75 is reproduced in Figure 4.45.

It was shown in Section 4.5.1 that the cause of filter divergence in run 40 could be traced to a single feature match. Unfortunately, a similar analysis of the features matched in the seconds leading up to the divergence of run 75 does not lead to a similar 'smoking gun'. Figure 4.46 shows the magnitude of the measurement residuals in the 10 seconds

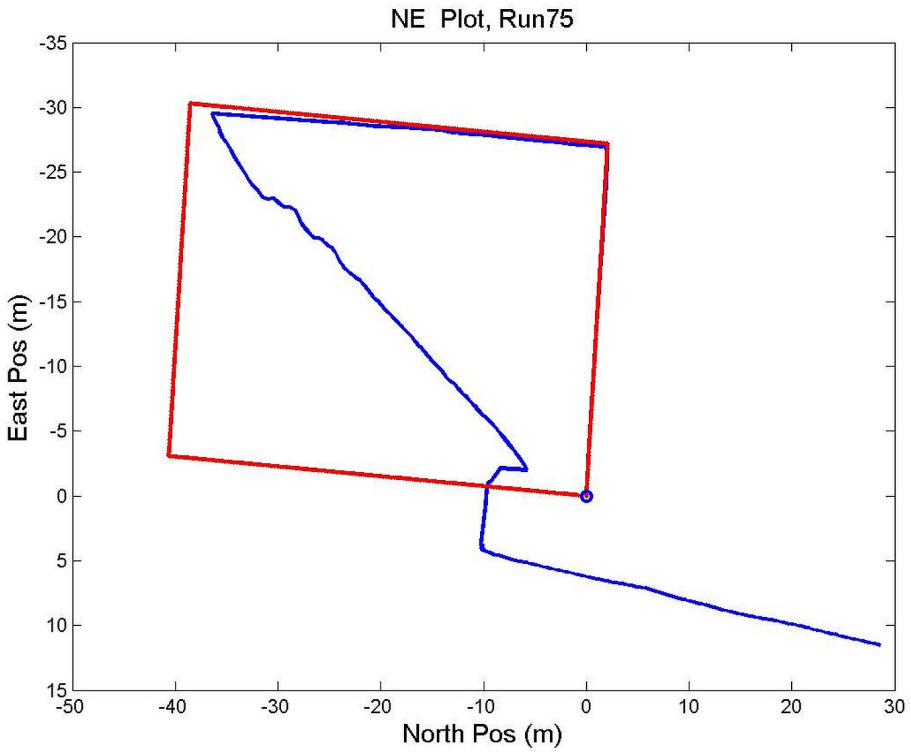

Figure 4.45: Run 75 N vs. E Plot, Diverged.

prior to the platform navigating through turn 2. It can be seen that there are a number of matches with residuals that are far larger in magnitude than those that lie within the assumed noise of 2 pixels.

A number of features result in bad matches, particularly in the area of the window in the right side of each picture contained within Figures 4.47 through 4.57. Similar errors are seen in some of the features matched in the area surrounding the nearest lighting panel. However, no egregious match similar to that found in run 40 is present. It is thus

hypothesized that the accumulation of errors from all of the spurious matches is to blame, and a remedy similar to that employed for run 40 is tried. All of the images shown in Figures 4.47 through 4.57 were removed, and the data reprocessed through the filter. Figure 4.58 shows the results of this process; again divergence is avoided by relying solely on the inertial measurements.

The downside is that a more robust residual monitoring scheme as proposed for run 40 would be unable to predict this divergent behavior. In fact, residuals of the magnitudes seen in Figure 4.46 commonly appear throughout all runs and times. It is unclear what the tipping point may have been in this case, but it has been shown that the cause certainly lies within the image updates used to correct the inertial drift.

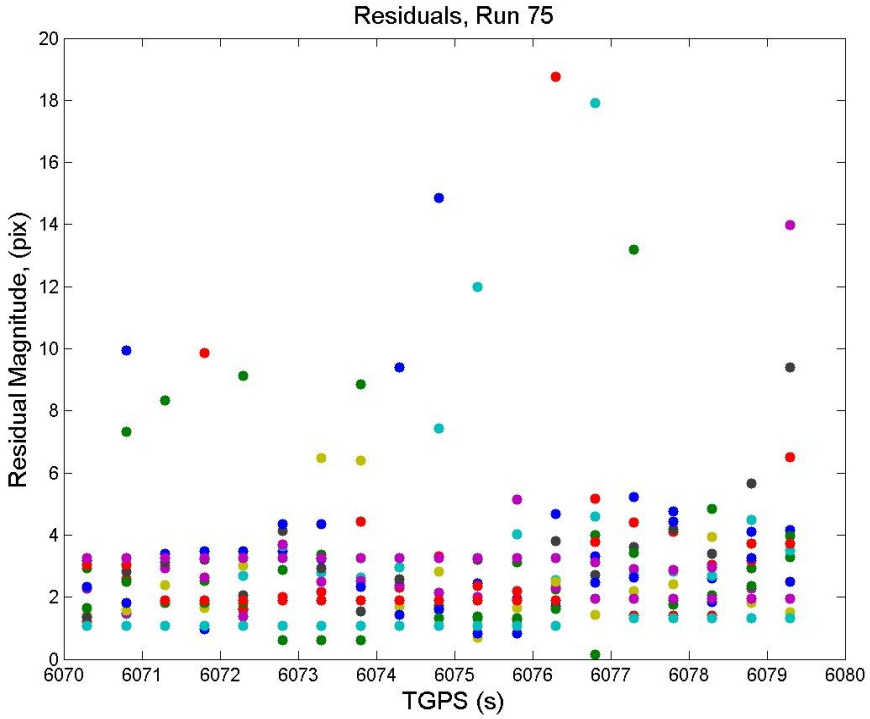

Figure 4.46: Run 75 Residuals.

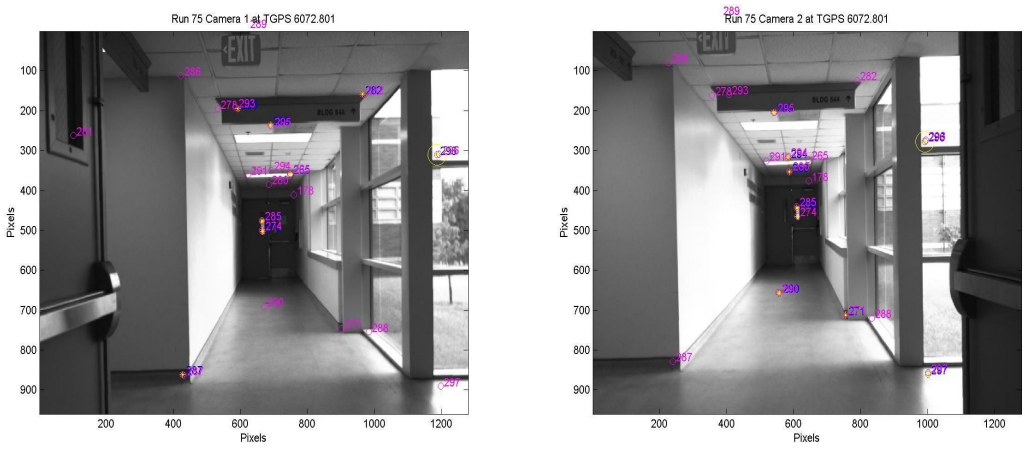

Figure 4.47: Run 75 Tracked Features, t = 6072.8 s.

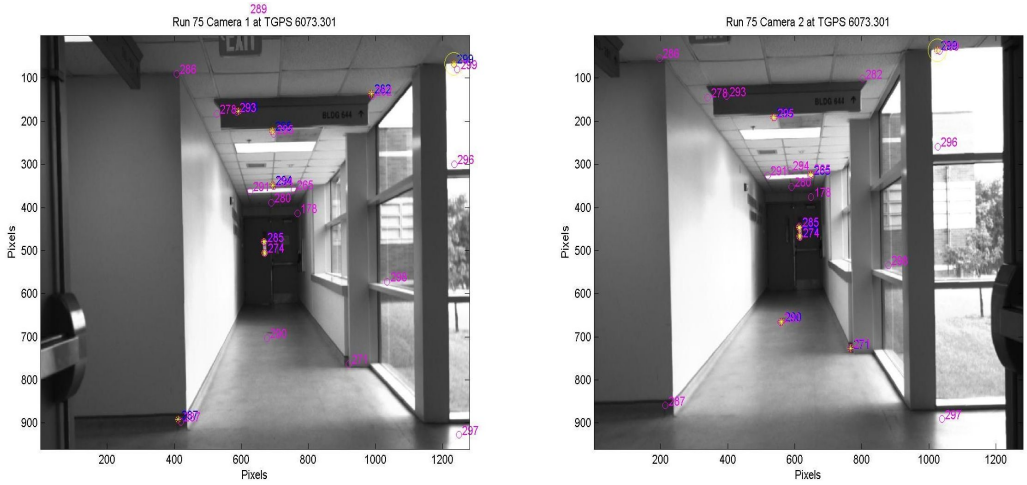

Figure 4.48: Run 75 Tracked Features, t = 6073.3 s.

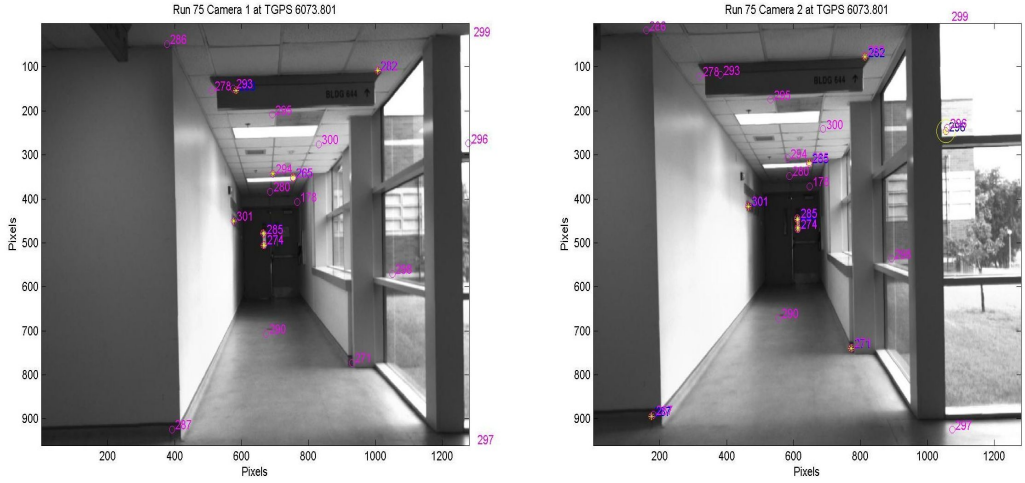

Figure 4.49: Run 75 Tracked Features, t = 6073.8 s.

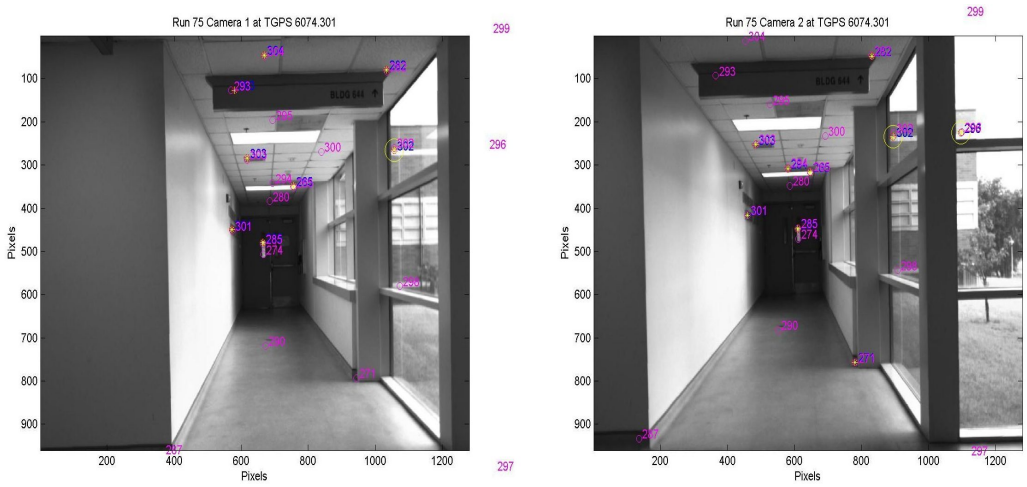

Figure 4.50: Run 75 Tracked Features, t = 6074.3 s.

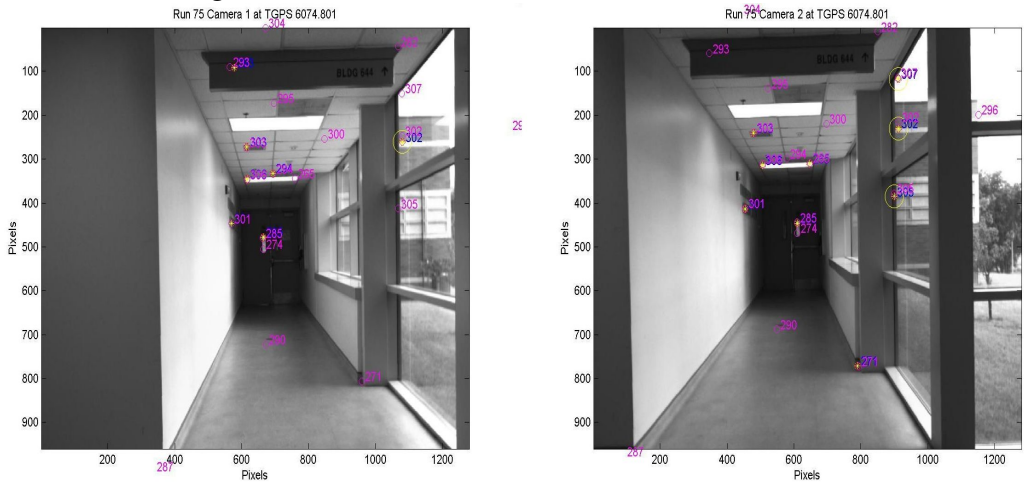

Figure 4.51: Run 75 Tracked Features, t = 6074.8 s.

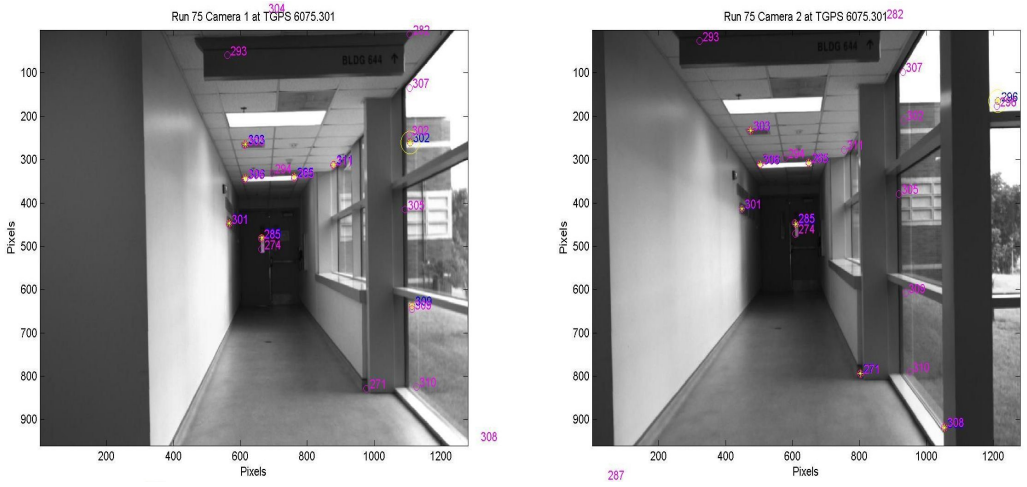

Figure 4.52: Run 75 Tracked Features, t = 6075.3 s.

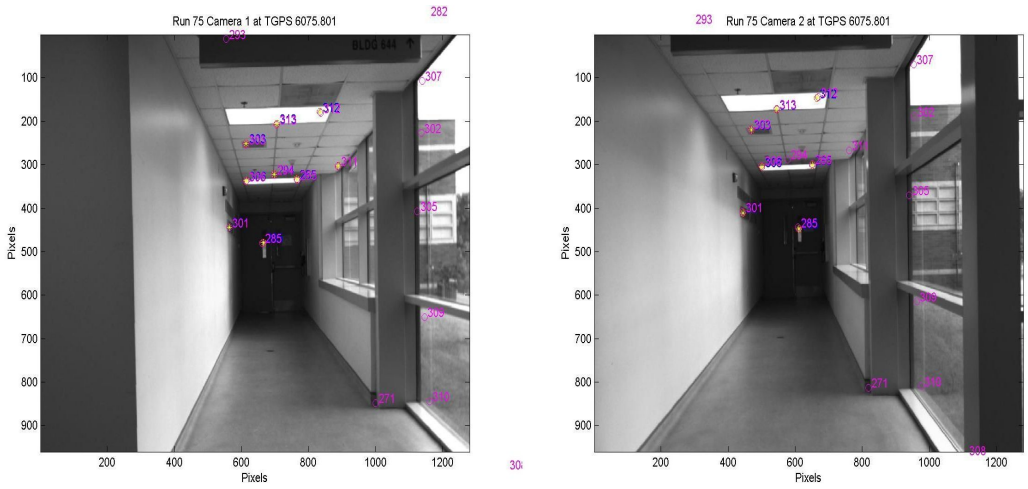

Figure 4.53: Run 75 Tracked Features, t = 6075.8 s.

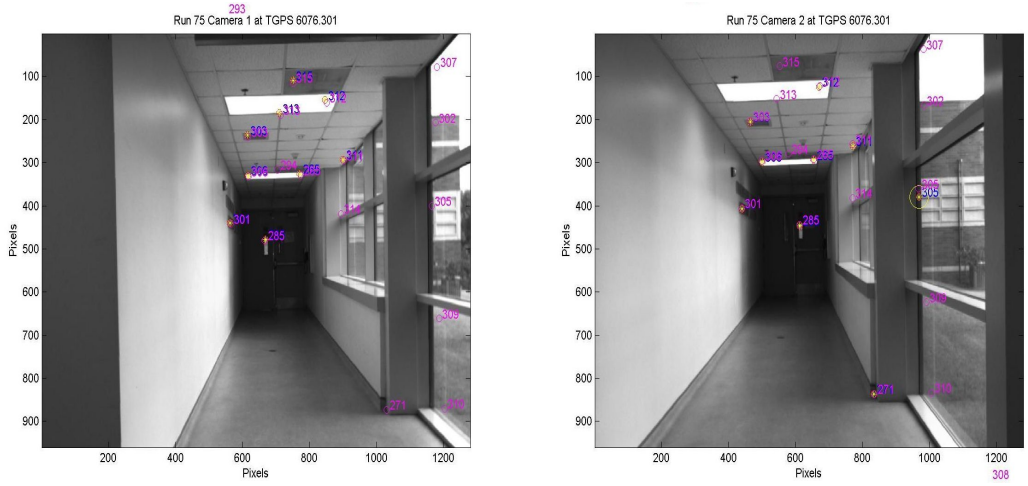

Figure 4.54: Run 75 Tracked Features, t = 6076.3 s.

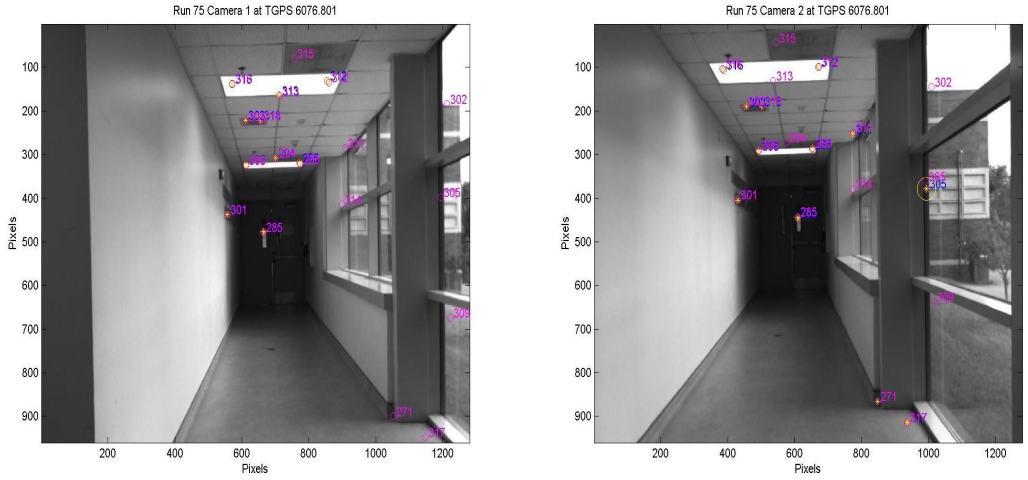

Figure 4.55: Run 75 Tracked Features, t = 6076.8 s.

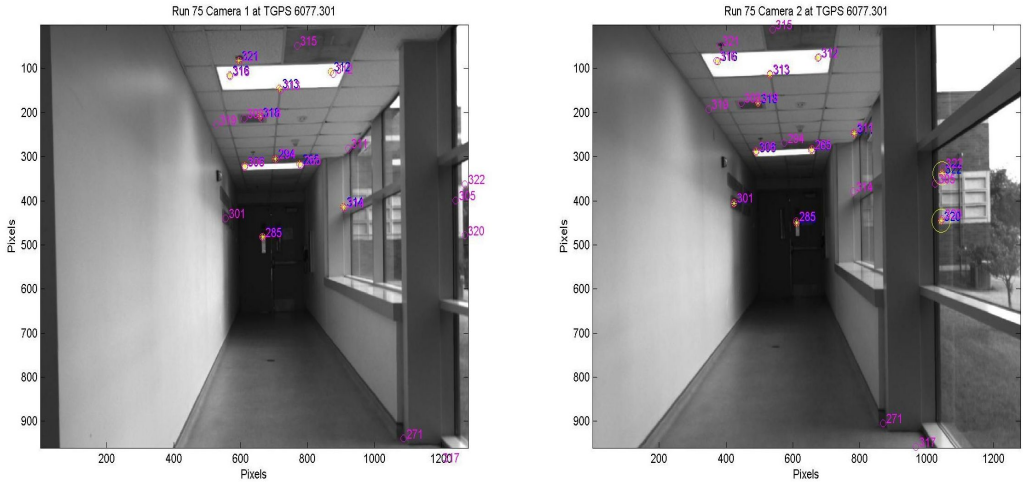

Figure 4.56: Run 75 Tracked Features, t = 6077.3 s.

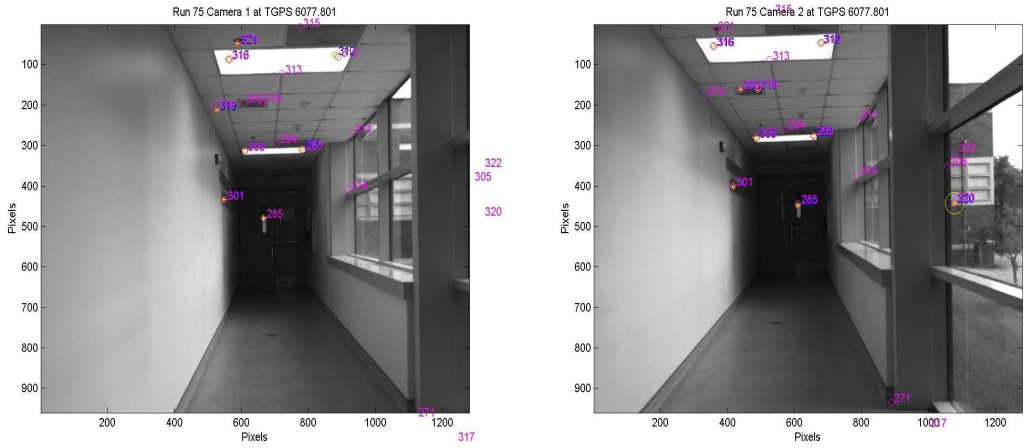

Figure 4.57: Run 75 Tracked Features, t = 6077.8 s.

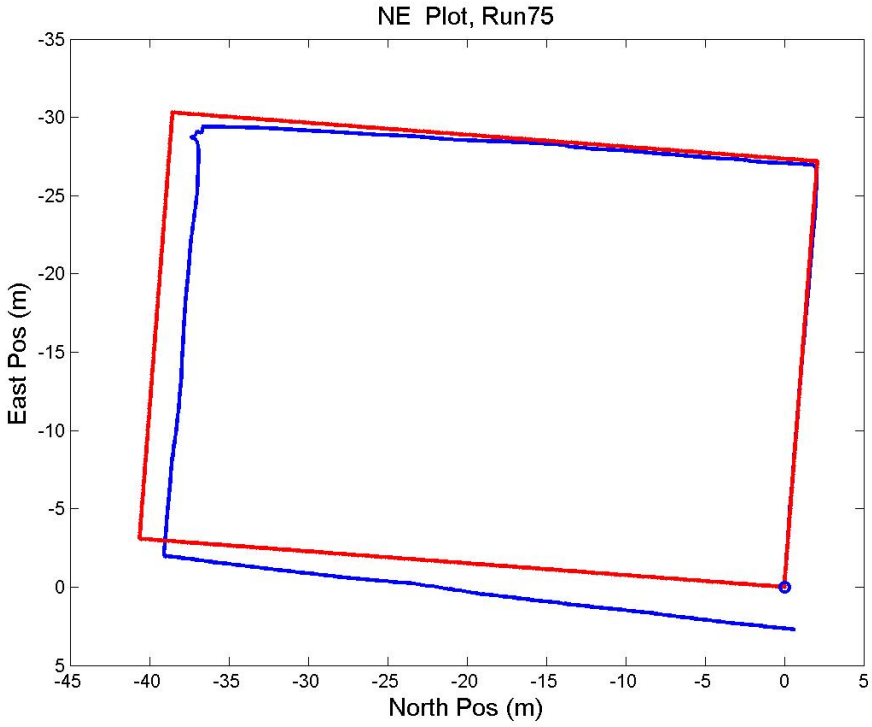

Figure 4.58: Run 75 NE Plot, Images Removed.

4.6 Summary

This chapter provided a full account of all of the data collected, as well as describing the approaches used in the characterization of IAN errors. The resulting graphs and statistics were presented along with a full analysis in each case; additionally, sources of errors were described and accounted for, when possible. Finally, the two divergent runs were investigated individually and causes of divergence found to be improper feature matching, with differing circumstances in each run. The information in this chapter represents a thorough and detailed characterization of IAN errors.

5 Conclusions and Future Work

This chapter concludes this document. Within is a summation of the conclusions that were drawn from the analysis presented in Chapter 4, as well as possible future research opportunities that may stem from this work.

5.1 Conclusions

Based upon the results presented in Chapter 4, a number of statements may be made regarding the performance of the filter under investigation, many of which confirm results presented in the various sources cited throughout the document. Concerning filter error distributions, it was shown that the position errors are not zero-mean. The North and East directions show biases which tend to the outside of the path in the two-dimensional plane, and these may have simply been the result of initial errors in placement of the platform, causing an improper estimation of the yaw bias. A strong bias trend was also seen in the Down direction, and a possible source of this error were presented in Section 4.2. Overall, the mean of the estimation error for the position of the platform was seen to be low, with a three-dimensional RMSE error peaking at 1.29 m. Given that no additional aiding was provided to the system, this is a pleasing result. Unfortunately, the magnitude of the errors seen in the attitude estimation were less impressive, averaging on the order of 3 mrad for roll and pitch, and even larger for the yaw. Given these results, it can be safely said that the characterization effort was a success. In general, the error magnitudes seen in the non-divergent cases comport with results presented in [8], [9] and [25], works which were based upon the same IAN algorithm. This agreement in results also serves as verification that the application of the IAN software and methods of analysis were correctly applied.

Regardless of the relatively low magnitude errors seen in the estimated means, the inaccuracies in the filter computed covariance provide what is likely the largest stumbling block to inclusion of this system's output in a larger navigation solution. Of all data types

considered, only the covariance estimated for the roll angle could be considered consistent, with the filter estimating 95% of the true covariance value when averaged across all runs and time. In particular, the position covariance estimates proved to be very inconsistent, with the filter estimating between the 10 to 20% range of the true covariance values. This confirms the findings of filter optimism reported in [1]; what is suprising, however, is how quickly optimism took hold. Vastly underestimated covariances (less than 50% of the true value) were typically seen within 100 timesteps, while based upon the findings in [7], it was expected that covariance optimism would not be seen until after hundereds of timesteps, if at all.

The error distributions and associated analysis reported in Chapter 4 only apply to the non-divergent runs. The divergent cases arise when something unexpected occurs within the filter, and it was shown that the filter used in this research exhibited a 2% divergence rate under the given conditions and chosen filter parameters. In addition, the particular causes of the divergent behavior were determined to the greatest extent possible; in both cases, divergence was seen to arise as a result of improper matches being made within the image processing algorithm. In run 40, it was shown that one particularly egregious matching error was to blame; in run 75, the cause was narrowed down to a subset of features matched over a period of 5 seconds. Blocking the suspect updates by removal of the corresponding images in each respective run resulted in an avoidance of the divergent behavior. The large magnitude of the measurement residual in run 40 appears to make residual monitoring an attractive candidate for early detection of divergent behavior; however, such a method would not have worked in the case of run 75, as the residual magnitudes seen immediately prior to divergence are similarly seen throughout all runs. More promise is seen in the filter computed covariance for these runs; the covariance given increases substantially after the divergence occurs. However, the effect does not

appear to be immediate enough to be used as a safeguard against accepting poor estimates just after the divergence has begun.

5.2 Future Work

Given that this work was performed to characterize the errors that occur in IAN applications using real-world data, the results presented are best viewed as a jumping-off point or reference for future works in this field, and particularly those that implement the stochastically constrained algorithm that was used here. It must be noted many of the results presented may be specific to the context of this data collection and the parameters supplied to the filter, especially the upward bias trend in the Down position estimates and the two cases of divergence. Although 100 runs represents a sizable data pool, a larger set, or longer run length, could be more informative. Additionally, it must be admitted that the truth source used represents a weakness in the experiment design; using high-integrity inertial data, or a GPS based truth source if the collection environment allows, would result in more accurate error analysis.

Regardless of the errors present in the truth, further analysis and use of this data is possible. Filter tuning could be investigated to try to improve the covariance estimates. A combination of residual and covariance monitoring could be implemented to detect filter divergence, or alternative feature matching algorithms could be developed to attempt to avoid it altogether. Finally, the data set could be used to investigate outlier-aware filtering. The error characteristics of the feature matches could be calculated to verify the parameters used in the distribution suggested in [20], and improvement in the filter covariance estimate compared to the results presented in this work.

Appendix A: Camera Calibration Parameters

Table A.1: Camera calibration parameters obtained through calibration, runs 1-41. Camera 125495 refers to the left camera in the collection rig, while 122865 was on the right.

	Cam. 125495	**Cam. 122865**	**Cam. 125495**	**Cam. 125886**
	Run: 1		**Runs: 2-7**	
Foc. Length	1354.9, 1354.9	1355.5, 1356.1	1358.5, 1358.5	1358.4, 1358.5
Princ. Point	667.95, 537.01	602.25, 504.70	667.69, 537.42	601.25, 503.19
k_1	-0.12604	-0.13914	-0.11723	-0.12769
k_2	0.15176	0.22382	0.13617	0.18925
k_3	-0.00017	0.00059	-0.0005	0.00034
k_4	0.00005	-0.00032	-0.00052	-0.00083
	Runs: 8-15		**Runs: 16-20**	
Foc. Length	1362.8, 1363.4	1363.6, 1364.6	1364.1, 1363.6	1364.9, 1366.0
Princ. Point	673.70, 538.33	599.60, 504.65	670.03, 541.47	601.41, 506.75
k_1	-0.11067	-0.12765	-0.11059	-0.13465
k_2	0.13888	0.1951	0.11539	0.21876
k_3	-0.0005	0.00031	-0.0008	0.00062
k_4	0.00013	-0.00037	-0.00047	-0.00037
	Runs: 21-23		**Runs: 24-41**	
Foc. Length	1356.5, 1356.9	1356.2, 1357.3	1355.8, 1356.0	1355.6, 1356.5
Princ. Point	666.58, 539.67	601.56, 506.87	667.97, 537.75	601.40, 504.82
k_1	-0.1166	-0.12962	-0.11784	-0.12836
k_2	0.13813	0.20376	0.14388	0.18533
k_3	-0.00053	0.00038	-0.00051	0.00055
k_4	-0.00051	-0.001	-0.00048	-0.00104

Table A.2: Camera Calibration Parameters, Runs 42-84.

	Cam. 125495	**Cam. 122865**	**Cam. 125495**	**Cam. 125886**
	Runs: 42-43		**Runs: 44-48**	
Foc. Length	1345.1, 1345.8	1344.6, 1346.2	1357.9, 1357.7	1354.8, 1356.6
Princ. Point	661.58, 535.63	598.82, 501.52	658.24, 533.52	596.88, 500.87
k_1	-0.10974	-0.12198	-0.12029	-0.13303
k_2	0.1398	0.18979	0.15229	0.19937
k_3	-0.00138	-0.00049	-0.00001	0.00096
k_4	-0.00138	0.0011	-0.00238	-0.00127
	Runs: 49-51		**Runs: 52-60**	
Foc. Length	1356.9, 1357.2	1356.0, 1357.1	1357.4, 1357.5	1357.2, 1358.0
Princ. Point	667.07, 541.34	596.88, 507.02	667.08, 538.98	601.69, 504.42
k_1	-0.11535	-0.12394	-0.11632	-0.12755
k_2	0.15844	0.19084	0.13919	0.18051
k_3	-0.00077	0.00033	-0.00046	0.0005
k_4	-0.00123	-0.00094	-0.00082	-0.00133
	Runs: 61-80		**Runs: 81-84**	
Foc. Length	1353.3, 1353.6	1353.0, 1353.9	1357.3, 1357.6	1359.6, 1360.9
Princ. Point	665.63, 540.64	602.98, 506.77	666.21, 538.69	594.95, 504.30
k_1	-0.11811	-0.12632	-0.11743	-0.12813
k_2	0.14648	0.17727	0.15759	0.20058
k_3	-0.00052	0.00053	-0.00067	0.00028
k_4	-0.00097	-0.00112	-0.00078	-0.00127

Table A.3: Camera Calibration Parameters, Runs 85-100.

	Cam. 125495	**Cam. 122865**	**Cam. 125495**	**Cam. 125886**
	Runs: 85-87		**Runs: 88-98**	
Foc. Length	1352.9,1353.2	1355.5, 1356.38	1367.4, 1367.9	1367.3, 1368.4
Princ. Point	667.67, 538.87	604.32, 504.60	670.02, 541.29	597.52, 508.42
k_1	-0.12155	-0.12666	-0.11536	-0.11907
k_2	0.16267	0.19115	0.16591	0.17960
k_3	-0.00068	0.00033	-0.00027	0.00073
k_4	-0.00056	-0.00075	-0.00020	-0.00095
	Runs: 99-100			
Foc. Length	1351.0, 1350.9	1352.02, 1352.8		
Princ. Point	662.55, 542.64	599.21, 504.11		
k_1	-0.12372	-0.13137		
k_2	0.16807	0.19626		
k_3	-0.00014	0.00020		
k_4	-0.00093	-0.00085		

Appendix B: Sensor Installation Parameters

Table B.1: Sensor Installation Parameters, Runs 1-23.

Left Translation and DCM		**Right Transalation and DCM**	
Run1			
$\begin{bmatrix} 0 \\ -226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0099 & 1 \\ 0.0117 & 0.9999 & 0.0099 \\ -0.9999 & 0.0117 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0005 & -0.0001 & 1 \\ 0.0139 & 0.9999 & 0.0001 \\ -0.9999 & 0.0139 & -0.0005 \end{bmatrix}$
Runs 2-7			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0096 & 1 \\ 0.0115 & 0.9999 & 0.0096 \\ -0.9999 & 0.0115 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0002 & 0.0021 & 1 \\ 0.0138 & 0.9999 & -0.002 \\ -0.9999 & 0.0138 & -0.002 \end{bmatrix}$
Runs 8-15			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0002 & -0.0136 & 0.9999 \\ 0.0115 & 0.998 & 0.0136 \\ -0.9999 & 0.0115 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0013 & 0.0013 & 1 \\ 0.0137 & 0.9999 & -0.0013 \\ -0.9999 & 0.0137 & -0.0013 \end{bmatrix}$
Runs 16-20			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0064 & 1 \\ 0.0110 & 0.9999 & 0.0064 \\ -0.9999 & 0.0110 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0021 & 0.0045 & 1 \\ 0.0131 & 0.9999 & -0.0045 \\ -0.9999 & 0.0131 & -0.0022 \end{bmatrix}$
Runs 21-23			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0103 & 0.9999 \\ 0.0114 & 0.9999 & 0.0103 \\ -0.9999 & 0.0114 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0016 & 0.0022 & 1 \\ 0.0136 & 0.9999 & -0.0022 \\ -0..9999 & 0.0136 & -0.0016 \end{bmatrix}$

Table B.2: Sensor Installation Parameters, Runs 24-60.

Left Translation and DCM		**Right Transalation and DCM**	
Runs 24-41			
$\begin{bmatrix} 0 \\ -226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0099 & 1 \\ -0.0119 & 0.9999 & 0.0099 \\ -0.9999 & 0.0119 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.001 & 0.0028 & 1 \\ 0.0140 & 0.9999 & -0.0028 \\ -0.9999 & 0.0140 & -0.0011 \end{bmatrix}$
Runs 42-43			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0103 & 0.9999 \\ 0.0110 & 0.9999 & 0.0103 \\ -0.9999 & 0.0110 & 0.0103 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0008 & -0.0030 & 1 \\ 0.0136 & 0.9999 & 0.0030 \\ -0.9999 & 0.0136 & -0.0008 \end{bmatrix}$
Runs 44-48			
$\begin{bmatrix} 0 \\ -226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0002 & -0.0139 & 0.9999 \\ 0.0114 & 0.9998 & 0.0139 \\ -0.9999 & 0.0114 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0005 & -0.0060 & 1 \\ 0.0137 & 0.9999 & 0.0060 \\ -0.9999 & 0.0137 & -0.0004 \end{bmatrix}$
Runs 49-51			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0132 & 0.9999 \\ 0.0111 & 0.9999 & 0.0132 \\ -0.9999 & 0.0111 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0015 & 0.0011 & 1 \\ 0.0133 & 0.9999 & -0.0011 \\ -0.9999 & 0.0133 & -0.0016 \end{bmatrix}$
Runs 52-60			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0090 & 1 \\ 0.0114 & 0.9999 & 0.009 \\ -0.9999 & 0.0114 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0016 & 0.0033 & 1 \\ 0.0136 & 0.9999 & -0.0033 \\ -0.9999 & 0.0136 & -0.0017 \end{bmatrix}$

Table B.3: Sensor Installation Parameters, Runs 61-100.

Left Translation and DCM		**Right Transalation and DCM**	
Runs 61-80			
$\begin{bmatrix} 0 \\ -226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0082 & 1 \\ 0.0114 & 0.9999 & 0.0082 \\ -0.9999 & 0.0114 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0014 & 0.0025 & 1 \\ 0.0136 & 0.9999 & -0.0024 \\ -0.9999 & 0.0136 & -0.0015 \end{bmatrix}$
Runs 81-84			
$\begin{bmatrix} 0 \\ -226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.0045 & 1 \\ 0.0083 & 1 & -0.0045 \\ -1 & 0.0083 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0010 & 0.0191 & 0.9998 \\ 0.0104 & 0.9998 & -0.0191 \\ -0.9999 & 0.0104 & -0.0012 \end{bmatrix}$
Runs 85-87			
$\begin{bmatrix} 0 \\ -226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0099 & 1 \\ 0.0112 & 0.9999 & 0.0099 \\ -0.9999 & 0.0112 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0014 & -0.001 & 1 \\ 0.0134 & 0.9999 & 0.001 \\ -0.9999 & 0.0134 & -0.0015 \end{bmatrix}$
Runs 88-98			
$\begin{bmatrix} 0 \\ -227.0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0001 & -0.0119 & 0.9999 \\ 0.0104 & 0.9999 & 0.0119 \\ -0.9999 & 0.0104 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 227.0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0018 & 0.0053 & 1 \\ 0.0125 & 0.9999 & -0.0053 \\ -0.9999 & 0.0124 & -0.0019 \end{bmatrix}$
Runs 99-100			
$\begin{bmatrix} 0 \\ -226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.001 & 1 \\ 0.0098 & 1 & 0.001 \\ -1 & 0.0098 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 226.2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.0055 & 0.0087 & 0.9999 \\ 0.0120 & 0.9999 & -0.0086 \\ -0.9999 & 0.0120 & -0.0056 \end{bmatrix}$

Bibliography

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

[2] Bar-Shalom, Y., X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley and Sons, New York, 2001.

[3] Bouguet, Jean-Yves. "Camera Calibration Toolbox for Matlab". http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed 1 April 2012.

[4] Committee, WGS 84 Development. *Department of Defense World Geodetic System 1984- Its Definition and Relationships with Local Geodetic Systems*. Technical Report TR8350.2, National Imagery and Mapping Agency, Bethesda, MD, 2000.

[5] Hartley, Richard and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, New York, 2^{nd} edition, 2003.

[6] Heikkilä, Janne and Olli Silvén. "A Four-Step Camera Calibration Procedure with Implicit Image Correction". *CVPR*, 1106–1112. 1997.

[7] Julier, Simon J. and Jeffrey K. Uhlmann. "A Counter Example to the Theory of Simultaneous Localization and Map Building". *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 4238–4243. May 2001.

[8] Jurado, Juan, Kenneth Fisher, and Michael Veth. "Inertial and Imaging Sensor Fusion for Image-Aided Navigation with Affine Distortion Prediction". *IEEE/ION Position, Location and Navigation Symposium (PLANS) 2012*, 518–526. Apr 2012.

[9] Jurado, Juan D. *Enhanced Image-Aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction*. Master's Thesis, Air Force Institute of Technology, 2012.

[10] Lee, Seok-Han, Tae-Eun Kim, and Jong-Soo Choi. "Correction of Radial Distortion Using a Planar Checkerboard Pattern and Its Image". *Consumer Electronics, 2009. ICCE '09. Digest of Technical Papers International Conference on*, 1 –2. January 2009.

[11] Lowe, D.G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[12] Madsen, K., H.B. Nielsen, and O. Tingleff. "Methods for Non-Linear Least Squares Problems". http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3215/pdf/imm3215.pdf, 2004. Accessed 9 April 2012.

[13] Marietta, Daniel A., Kenneth Fisher, and Clark Taylor. "Monte Carlo Error Characterization of EKF-Based Image Aided Navigation". *ION International Technical Meeting (ITM) 2013*. Jan 2013. Publication pending.

[14] Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 1*. Navtech, Arlington, Virginia, 2001. Republished.

[15] Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 2*. Navtech, Arlington, Virginia, 2001. Republished.

[16] Prahl, Dayvid M. *Coupling Vanishing Point Tracking with Inertial Navigation to Estimate Attitude in a Structured Environment*. Master's Thesis, Air Force Institute of Technology, 2011.

[17] Shanmugan, K. and A. Breipohl. *Random Signals: Detection, Estimation and Data Analysis*. Wiley, New York, 1988.

[18] Stern, Julio M. "Spencer-Brown vs. Probability and Statistics: Entropy's Testimony on Subjective and Objective Randomness". *Information*, 2(2):277–301, June 2011.

[19] Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Springer, New York, 1st edition, November 2010.

[20] Taylor, Clark N. "Improved Fusion of Visual Measurements Through Explicit Modeling of Outliers". *IEEE/ION Position, Location and Navigation Symposium (PLANS), 2012*, 512–517. April 2012.

[21] Titterton, D.H. and J.L. Weston. *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics, Reston Virginia, 2nd edition, 2004.

[22] Van Loan, Charles F. "Computing Integrals Involving the Matrix Exponential". *IEEE Transactions on Automatic Control*, AC-23(3):395–404, June 1978.

[23] Veth, M. and J. Raquet. "Fusion of Low-Cost Imaging and Inertial Sensors for Navigation". *Proceedings of the ION 19th International Technical Meeting of the Satellite Division, ION GNSS 2006*, 1093–1103. Sep 2006.

[24] Veth, M., J. Raquet, and M. Pachter. "Stochastic Constraints for Efficient Image Correspondence Search". *IEEE Transactions on Aerospace and Electronic Systems*, 42(3):973–982, 2006.

[25] Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. Thesis, Air Force Institute of Technology, 2006.

[26] Zhang, Zhengyou. "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations". *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, 666–673. 1999.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22-03-2013	Master's Thesis	Apr 2012 – Mar 2013

4. TITLE AND SUBTITLE

Error Characterization of Vision-Aided Navigation Systems

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Marietta, Daniel A, Civ AD-21, USAF

5d. PROJECT NUMBER
12-335; 12-370

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENY)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-13-M-33

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Clark N. Taylor, DR III, Senior Electronics Engineer,
AFRL/RY Sensors Directorate
2241 Avionics Circle, Bldg 600
Wright Patterson AFB OH 45433-7318
clark.taylor.3@us.af.mil
DSN 798-8148

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RY

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The goal of this work is to characterize the errors committed by an Image Aided Navigation (IAN) algorithm that has been developed for use as a navigation tool in GPS denied areas. The filter under study was developed by the Air Force Institute of Technology's Advanced Navigation Technology center, and has been the focus of numerous research efforts. Unfortunately, these studies have all been based on single runs or simulations, and such results may not be indicative of the true filter performance. This problem extends to IAN publications in general; no analysis of IAN based upon a sizable real world data collection appears in the literature. This issue is addressed by applying Monte Carlo analysis methods to a 100 run data set collected using a joystick controlled robot outfitted with an inertial unit and stereo cameras. The averaged error magnitudes are found to be within 1 m RMSE. In addition, optimism in the filter computed covariance is verified. Finally, two instances of filter divergence are explored, with the causes being traced to feature matching errors. The results of this work will support future research efforts by providing a baseline measure of filter performance against which prospective enhancements may be compared.

15. SUBJECT TERMS

Image aided navigation, visual odometry, feature tracking, feature matching, extended Kalman filter

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	145	Lt Col Kenneth A Fisher, USAF (ENG)
U	U	U			19b. TELEPHONE NUMBER (Include Area Code) (937)255-3636, ext 4677