EXPERIMENTAL CELESTIAL NAVIGATION PROTOTYPE FOR COMPUTER VISION BASED NAUTICAL SIGHT REDUCTION

BY

THOMAS C.K. FULLER

B.S. in Electromechanical Engineering, Wentworth Institute of Technology,
Boston, 2013

THESIS

Submitted to the University of New Hampshire
In Partial Fulfillment of the Requirements for the Degree of
Master of Science
in
Mechanical Engineering

May, 2016

EXPERIMENTAL CELESTIAL NAVIGATION PROTOTYPE FOR COMPUTER VISION BASED NAUTICAL SIGHT REDUCTION

BY

THOMAS C.K. FULLER

This thesis has been examined and approved.

Thesis Director, Dr. May-Win L. Thein,

Associate Professor of Mechanical Engineering and Ocean Engineering

---

Dr. Barry Fussell,

Professor of Mechanical Engineering

Dr. Michael Carter,

Associate Professor of Electrical & Computer Engineering

Date

Dedication

Dedicated to Craig Curtis, role model and friend.

Acknowledgments

This research was made possible by the contributions of many people. First thanks to Prof. May-Win Thein, for advising this project since August 2013 and affording me many opportunities over my time at UNH. Thank you to Prof. Barry Fussell and Prof. Michael Carter for serving on the thesis committee.

Christopher Hashem and I have been studying together for the last eight years of undergraduate and graduate school. That's important for me to put here.

To Mike Johnson and Shuai Chen, this work wouldn't have been possible without your contributions in the optimization techniques and the design of the experimental setup.

This research was greatly aided through the many conversations I had with William Nitsch about image processing and machine learning.

Thank you to the Mechanical Engineering Department for supporting me through assistantship during my time at UNH.

Lastly I'd like to thank my friend, Michael Tian, my mom, Mimi Wong and my dad, John Fuller for encouraging me to set high, even lunar goals for myself.

Contents

List of Figures

List of Tables

ABSTRACT

EXPERIMENTAL CELESTIAL NAVIGATION PROTOTYPE FOR COMPUTER VISION BASED NAUTICAL SIGHT REDUCTION

by

Thomas C.K. Fuller

University of New Hampshire, May 2016

This thesis presents the development and analysis of an experimental platform for celestial navigation using the Directly Computed Nautical Sight Reduction Algorithm. The goal is to obtain an estimate of latitude and longitude using an image of the stars and measurements from an IMU as an input. Analytical simulation results indicate that the presented method is viable for navigation for three different observational formats and a wide range of camera pointing directions. Experimental development is presented including cost function analysis for minimizing the error in the position estimate. The results of the optimization were that an error of 0.0585°, approximately 3 nautical miles, was achieved. The presented device is also compared to other established methods of celestial navigation.

Chapter 1

Introduction

1.1 Methods of Celestial Navigation

Lander and rover type vehicles have been a primary source of scientific insight when exploring the surface of another planet. There has been a trend of surface exploration missions from intentional hard impact, to soft lander, and finally to the modern rover. Each of these exploration techniques has improved scientific output of the surface exploring apparatus. The first extraterrestrial landing was performed by *Luna 2* [1]. This device was intentionally impacted on the surface of the moon. While it was able to determine some interesting properties of the magnetic field and Van Allen belt, due to the strong impact force of the probe, very little information was revealed about target surface, that of the moon. Soft landing, being a requirement for manned lunar landing, was an area of active research during the 1960s and was first achieved on the Moon by *Surveyor 1* on 30-May-1966. This probe delivered important scientific results including data on bearing strength, temperatures, and radar reflectivity of the moon [2]. It can be seen from Fig. 1.1 that data transmission rates for space systems will only increase in the future.

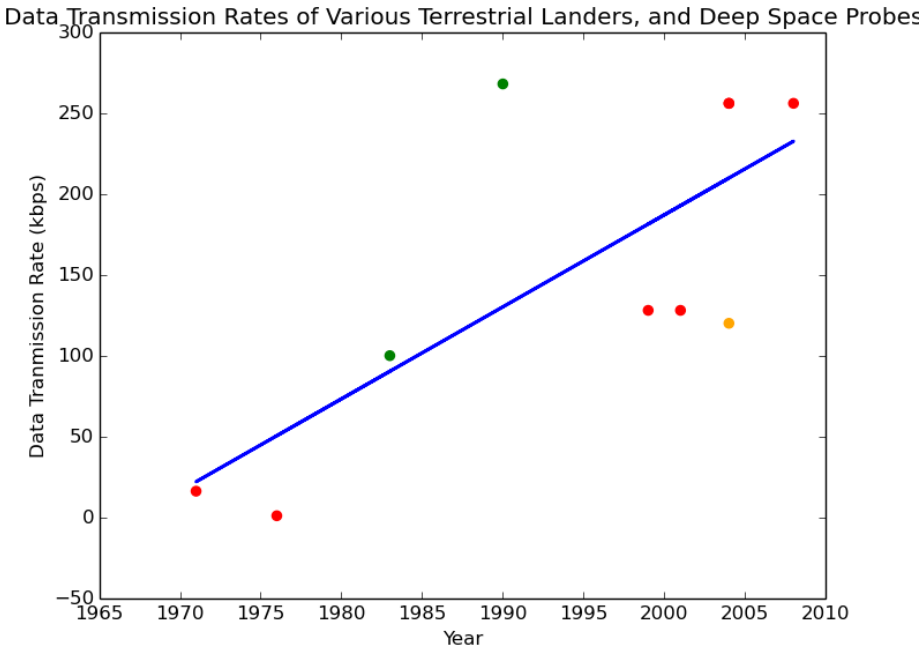

Figure 1.1: Data Transmission Rates for Various Extraterrestrial Probes. Line of Best fit included. Different colors indicate different destination planets[3],[4],[5]

This greater data rate necessitates better functionality from extraterrestrial mobility solutions. Navigational capability has ceased to be a fancy, non-required feature. It is greatly increasing the scientific yield of exploratory rovers. Improvement of such techniques will only serve to increase the amount of science yield.

Different methods of extraterrestrial navigation have been performed. Due to the prohibitive nature of implementing a GPS constellation in orbit around interesting celestial bodies, these methods all have some celestial navigation component. The methods surveyed here are the Deep Space Network, Phobos Transit Method, Attitude Matrix Formulated Method, Modi-

fied Attitude Matrix Formulated Method, and the Nautical Sight Reduction Method.

Deep Space Network

The Deep Space Network is a comprehensive communications network used for communication with current deep space probes. The Deep Space Network uses light from a quasar in conjunction with three ground stations located in Barstow, California, Madrid, Spain, and Canberra, Australia[6]. Deep Space Network is extremely accurate, leading to position error circles between 5 and 10 meters for certain applications[7]. This level of accuracy is possible over a widespread variety of applications from surface exploration to deep space flybys. The primary limitations of Deep Space Network are that it has a geometric weakness in the southern declinations and that it requires a precisely timed interaction between the ground station, a quasar, and the vehicle it is implemented on[8]. The device presented in this research is unlikely to achieve this accuracy on the surface of another planet. It is possible to implement an experimentally feasible, self-contained celestial navigation device which does not require complicated infrastructure.

Phobos Transit Method

The Phobos Transit Method is a method that relies on visual observations of the martian moon Phobos for navigating the Martian surface. The Phobos transit method has a theoretical accuracy of approximately 5000m and it may be possible to reduce that error to approximately 2000m. It is theoretically possible that by using detailed models of interesting celestial bodies, with the intention of more accurately estimating sidereal time, and therefore local hour angle, that this method could be applicable to any celestial body with

a sattelite whose properties of motion are known. As of yet, this method has not been verified experimentally by rovers on the surface of Mars[9]. Further analysis on on the calculation of sidereal time, nutation, precession, and abberation effects for Martian celestial navigation was examined by Malay et. al [10]. It should also be noted that this method has great potential for self containment, and would be possible to implement without a supporting satellite or communications network.

Attitude Matrix Formulated Method

The attitude matrix formulated method was first developed by Samaan et. al[11]. This self-contained method uses attitude matrices as well as a star imaging software based on the Pyramid-LISA algorithm. This method was brought to experimental life as the Compass Star Tracker. The Compass Star Tracker was relatively successful as a proof of concept device. The attitude matrix formulated method obtained an experimental accuracy of 61.23 miles, or 98.54km[12]. The primary disadvantage of this method is that the camera optical axis must be aligned with the zenith to ensure maximum accuracy. Attempts to reduce or eliminate this zenith orientation requirement did not seem to be successful. Chapter 2 will discuss this method in much greater detail.

Modified Attitude Matrix Formulated Method

The modified attitude matrix formulated method was developed by Thein et al. It is essentially the attitude matrix formulated method, with an emphasis on misalignment and sensor implementation. This allowed interesting analytical calculations of the accuracy to be computed. It was found that for an accuracy of 50m in this implementation, the combined errors of all orthog-

onal misalignment needed to be less than 5.93 arcseconds[13]. One major concern in the past research is experimental verification of this method. Experimental results were found using a backtracking method[14]. This method uses the current position as an input, and is used to show that a valid star tracker and inertial measurement unit rotations could be obtained. Correct celestial navigation must occur in reverse of this backtracking process. The position must be the output, and an observation of the celestial bodies must be an input.

Nautical Sight Reduction Method

The method of celestial navigation used by this thesis is called Nautical Sight Reduction, detailed in the Nautical Almanac[16]. The presented implementation uses a computer vision technique to observe three or more stars. The primary concept is that an imaginary line could be drawn from the star through the center of the Earth. The point where that line intersects the surface of the earth is called the ground position. Then, if the angle from the horizon to the celestial body could be measured, a locus of points where that angle is constant could be drawn. This locus of points is called a circle of position. The center of the circle of position is the ground position of the observed star. Three or more circles of position are intersected to determine an estimate of the observer's position. This concept is illustrated in Fig. 1.2

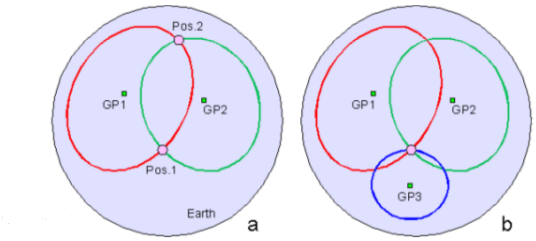

Figure 1.2: Diagram of Nautical Sight Reduction Method[15]

1.2 Research on Extra-Terrestrial Surface Navigation

One goal of the University of New Hampshire Advanced Controls Laboratory is a complete Guidance, Navigation, and Control solution for navigation on celestial bodies. This solution is represented by the block diagram of Fig. 1.3 On the part of Guidance, work performed by Johnson and Thein exam-

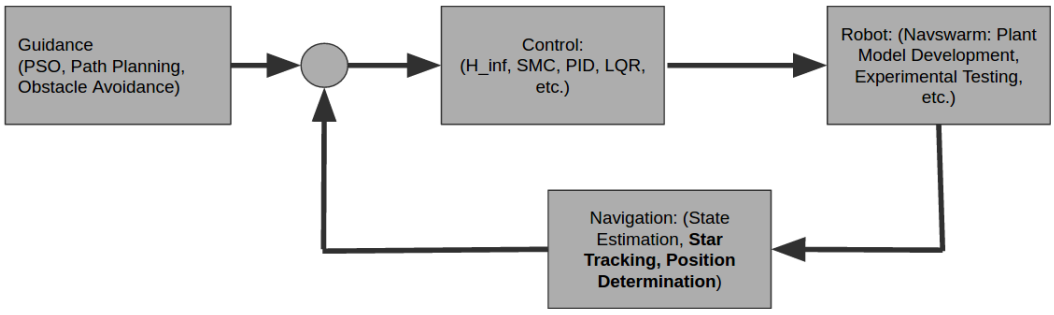

Figure 1.3: Complete Guidance, Navigation, and Control Solution for Extraterrestrial Surface Navigation. The material discussed in this thesis are bolded in the feedback block.[17]

ined the use of the Particle Swarm Algorithm, obstacle avoidance, and path planning[18]. Particle Swarm is a distributed optimization algorithm being used in a novel way; It optimizing its cost function will be used for extrac-

tion of natural resources. The additional features will be to this system, because they allow generic rovers, known as mobility solutions to traverse dangerous paths. One important part of this work is the senior project team ET-Navswarm. The Navswarm is a fleet of autonomous rovers on which the guidance, navigation, and controls will be experimentally verified.

On the part of Controls, work performed by Underwood and Thein used a differential-drive mobility solution with the same plant model as the Navswarm rover[19]. Their study surveyed the use various control algorithms for vehicle motion. These controllers included Proportional Integral Differential (PID) control, Linear Quadratic Regulator (LQR) control, H_∞ Optimal Control, Fuzzy-LQR, and Sliding Mode Control (SMC). These controllers evaulated on the criteria of efficiency, accuracy, speed of response, controller effort, actuator effort, steady state error, robustness to noise and disturbances, computational effort, and ease of implementation. This work concluded that given enough computational power, the optimal H_∞ controller would be the controller of choice for the differential drive mobility solution. If computational power is a scarce resource for the Navswarm rover, then SMC is the recommended controller, due to its performance compared to its ease of implementation. It is worth noting that of the examined controllers, SMC had the highest amount of control effort.

On the part of Navigation, work done by Perkins and Thein examined the use of a matrix formulated algorithm for use in celestial navigation. This is a modification of an algorithm originally proposed by Samaan et. al[11]. Perkins and Thein examined the use of several feedback estimation techniques for the mobility solution. These estimators included Extended Kalman Filter (EKF), Sliding Mode Observer (SMO), and H_∞ Observer. For ease of implementation these observers were simulated on the mobility solution with

a PID controller. Perkins concluded, similarly to Underwood, that the H_∞ optimal observer method drove the state error closest to zero, and that SMO had the least computational time. Monte Carlo analysis was performed to study the position based error. In that study it was found that a angular error in longitude was a function of latitude, and that the error decreased as the estimate was closer to the poles. The was an expected result because lines of longitude converge at the pole.

1.3 Thesis Contributions

The primary objective of this thesis is to provide the theory and proof-of-concept experimental platform for a self-contained device that can use an image of the stars and output meaningful position data. This would eliminate the need for the backtracking method performed by Perkins and Underwood. A secondary objective is to demonstrate the ability of this device to use a wide range of zenith angles. This would be an improvement over the Compass Star Tracker, which needed to be as close to zenith oriented as possible [12]. The tertiary objective are to provide some analysis on the performance of the individual sensors and overall accuracy of the device. This work serves as the first step to achieving an experimental platform for dynamic control of an extraterrestrial robotic mobility solution using celestial navigation.

1.4 Thesis Outline

Chapter 1: Introduction Introduces the problem statement, background research, and the thesis outline.

Chapter 2: Celestial Navigation Algorithms This chapter discusses

the matrix formulated celestial navigation algorithm. Its properties, and the decision to change algorithms are discussed. The new algorithm is proposed. The method for locating the stars in an image is the Blind Astrometric Calibration Method. The method for using the star locations to determine location is called the Sight Reduction method. These two methods are discussed separately, then their integration into one navigational algorithm is proposed.

Chapter 3: Simulation of Nautical Sight Reduction Algorithm Simulations are presented in order to determine the validity of the new algorithm. Tests performed include observational format, accuracy, and uncertainty and error analysis.

Chapter 4: Experimental Design and Verification The developed experimental solution is discussed along with experimental results and analysis on accuracy, uncertainty, and sensor communication. A generic optimization method of calibrating unknown pitch and roll misalignment is proposed and preliminary results are presented.

Chapter 5: Discussion This section discusses the results of the theoretical derivations, the simulations, and the experiments presented in the previous chapters. This section discusses present successes and failures and makes suggestions on future works.

Chapter 6: Conclusions The thesis conclusions are made, relating the objectives set forth in Chapter 1 to the progress that was made throughout the thesis.

Chapter 2

Celestial Navigation Algorithms

In this chapter, three algorithms are discussed in detail. The first one is the matrix formulated algorithm developed by et. al[11]. The second is a modified matrix formulated method developed by Thein et. al[13]. The third is the presented algorithm of this thesis, which is based on Nautical Sight Reduction.

2.1 Attitude Matrix Formulated Method (Compass Star Tracker)

The Attitude Matrix Formulated method, which in this work will also be referred to as the Compass Star Tracker, determines the relative attitudes of four different reference frames. The references frames will be denoted by $A^{X/Y}$ to indicate that this matrix rotates the attitude from X to Y coordinate frames.

Four coordinate frames are defined in this research are identical to those

defined in the Compass Star Tracker. They are denoted by the following letters. B is the body or camera reference frame. I is the inertial, geocentric reference frame. G is the Greenwich reference frame. and L is the local reference frame.

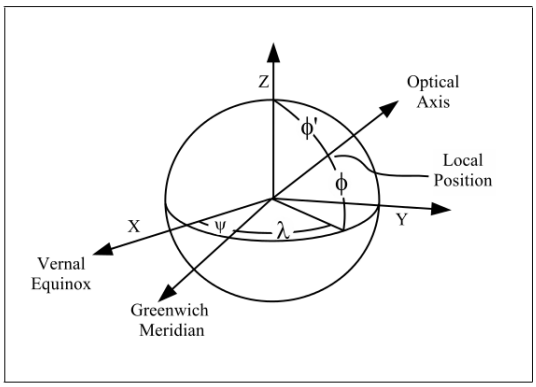

Figure 2.1: Inertial Reference Frame for Celestial Navigation

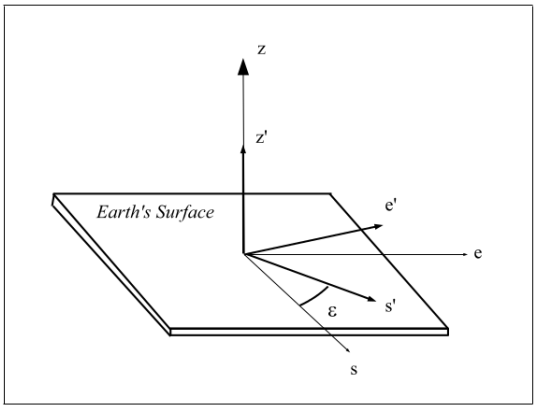

Figure 2.2: Local Reference Frame for Celestial Navigation

The notation is also largely the same as with the Compass Star Tracker,

- $\psi(t)$ is the angle between the vernal equinox and the Greenwich Meridian.

- λ is the longitude
- ϕ is the latitude
- $\phi^{'}$ is the compliment of latitude ϕ.
- $\vec{oa}$ is the pointing direction of the camera. In Fig. 2.2, this is denoted as $z^{'}$

In the original attitude matrix formulation (Compass Star Tracker), the camera optical axis was assumed to be aligned with the zenith direction. This work provides some experimental confirmation that that is not necessary. Given such a zenith requirement, the attitude transformation between the body and local coordinate frames are given as

$$A^{B/L} = \begin{bmatrix} cos(\epsilon) & sin(\epsilon) & 0 \\ -sin(\epsilon) & cos(\epsilon) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Eq.(2.1) is where the CST gets the "Compass" part of its name. Later on we will see that the derivation of $A^{B/L}$, called Γ in the modified attitude matrix formulation, so simple if the CST is at a non-zenith pointing orientation. Given a time invariant system, the rest of the attitude matrices are straightforward to derive. The attitude transformation between the local reference frame and the Greenwich frame are given by

$$A^{L/G} = R_2(\phi^{'})R_3(\lambda) = \begin{bmatrix} cos(\phi^{'}) & 0 & -sin(\phi^{'}) \\ 0 & 1 & 0 \\ sin(\phi^{'}) & 0 & cos(\phi^{'}) \end{bmatrix} \begin{bmatrix} cos(\lambda) & sin(\lambda) & 0 \\ -sin(\lambda) & cos(\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$A^{L/G} = \begin{bmatrix} cos(\phi^{'})cos(\lambda) & cos(\phi^{'})sin(\lambda) & -sin(\lambda) \\ -sin(\lambda) & cos(\lambda) & 0 \\ sin(\phi^{'})cos(\lambda) & sin(\phi^{'})sin(\lambda) & cos(\phi^{'}) \end{bmatrix} \tag{2.3}$$

The next attitude that is transformed about is from Greenwich to Inertial Frame. To do this many interesting factors are considered. These exact factors are given in a later section of this chapter. In a simplified version of this equation, the three unknown variables for position determination are ϕ, λ, and ϵ. These parameters can be solved as follows.

$$\Big[A^{B/L}A^{L/G} = A^{B/I}A^{I/G} = A\Big] \tag{2.4}$$

The product of $A^{B/I}$ and $A^{G/I}$ is known because $A^{B/I}$ is determined from from the star tracking and attitude determination algorithms. All the unknown parameters are contained in the left side of the given equation. This equation can be given as

$$A^{B/L}A^{L/G} = \begin{bmatrix} cos(\epsilon) & sin(\epsilon) & 0 \\ -sin(\epsilon) & cos(\epsilon) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\phi^{'})cos(\lambda) & cos(\phi^{'})sin(\lambda) & -sin(\lambda) \\ -sin(\lambda) & cos(\lambda) & 0 \\ sin(\phi^{'})cos(\lambda) & sin(\phi^{'})sin(\lambda) & cos(\phi^{'}) \end{bmatrix} \tag{2.5}$$

$$A^{B/L}A^{L/G} = \begin{bmatrix} cos(\epsilon)cos(\phi^{'})cos(\lambda) - sin(\epsilon)sin(\lambda) & cos(\epsilon)cos(\phi^{'})sin(\lambda) + sin(\epsilon) & -cos(\epsilon)sin(\phi^{'}) \\ -sin(\epsilon)cos(\phi^{'})cos(\lambda) - cos(\epsilon)sin(\lambda) & -sin(\epsilon)cos(\phi^{'})sin(\lambda) + cos(\epsilon)cos(\lambda) & sin(\epsilon)sin(\phi^{'}) \\ sin(\phi^{'})cos(\lambda) & sin(\phi^{'})sin(\lambda) & cos(\phi^{'}) \end{bmatrix} \tag{2.6}$$

From the above matrix, called Φ in some literature, λ, ϵ, and ϕ can be obtained.

$$\lambda = atan2[A(3,2), A(3,1)] \tag{2.7}$$

$$\epsilon = atan2[A(2,3), -A(1,3)] \tag{2.8}$$

Where the atan2 function deals with ambiguities related to what quadrant its arguments are in. It is given as

$$\text{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{2.9}$$

The atan2 function takes the arctangent and avoids issues based on the quadrant x and y are in.

The latitude $\phi^{'}$ is given as

$$\phi = 90 - \frac{A(3,2)}{sin(\lambda)} \tag{2.10}$$

Modified Attitude Matrix Formulated Method

The modified attitude matrix is the same general concept of changing attitudes, and shown below in Eq (2.11).

$$[H(\epsilon)]^{Down}_{NED}[U]^{NED}_{ENU}[LL(\lambda,\phi)]^{ENU}_{SD} = [\Phi] = [\Gamma]^{Down}_{A}[M_A]^{A}_{Body}([\Omega]^{SD}_{Inertial}[\Delta]^{Inertial}_{C}[M_C]^{C}_{Body})^T \tag{2.11}$$

Eq.(2.11) performs the same rotation matrices as the Compass Star Tracker algorithm. The right hand side of this equation describes the accelerometer (Γ) matrix and the star tracker (Δ) matrix. M_A and M_C are the respective misalignments. Ω

is the planetary (lunar) model. This allows simple analysis of misalignments between the accelerometer and body frame in M_A, and the star tracker and inertial frame in M_C[13]. In the next section, we will see some analysis of the Γ and Δ matrices which led to the selection of the current algorithm. The first is an analysis of the modified attitude matrix version of Γ and presents a more efficient, correct method by which the body to local coordinate transformation can be performed.

2.2 Star Tracker Matrix, Δ for Modified Attitude Matrix Method

The modified attitude matrix formulation showed that the Δ matrix begins with the star tracker quaternion.

$$StarTracker_{measured} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_0 \end{bmatrix} = \begin{bmatrix} sin(\frac{\theta}{2})cos(\beta_x) \\ sin(\frac{\theta}{2})cos(\beta_y) \\ sin(\frac{\theta}{2})cos(\beta_z) \\ cos(\frac{\theta}{2}) \end{bmatrix} \tag{2.12}$$

The quaternion values were assembled into a 3x3 matrix with the following format

$$M = \begin{bmatrix} Q_1 & Q_2 & Q_3 \end{bmatrix} \tag{2.13}$$

where,

$$Q_1 = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_0^2 \\ 2(q_1q_2 - q_3q_4) \\ 2(q_1q_2 + q_3q_4) \end{bmatrix} \tag{2.14}$$

$$Q_2 = \begin{bmatrix} 2(q_1q_2 + q_3q_4) \\ -q_1^2 - q_2^2 - q_3^2 + q_4^2 \\ 2(q_1q_2 - q_3q_4) \end{bmatrix} \tag{2.15}$$

$$Q_1 = \begin{bmatrix} 2(q_1q_2 - q_3q_4) \\ 2(q_1q_2 + q_3q_4) \\ -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.16}$$

The derivations of the modified attitude matrix formulated method did not provide a detailed relationship between an observation the stars and this quaternion/rotation matrix. The original attitude matrix formulation used an algorithm called Pyramid-LISA[21]. In the presented research, a similar method called the Blind Astrometric Method will be used to address that the star tracker measurement should obtained using an observation of the stars, as opposed to the backtracker method.

2.3 Body to Local Rotation Matrix, Γ

The accelerometer rotation matrix Γ rotates from the body to local coordinate frames. In some literature it is denoted as $A^{B/L}$. Experimental implementation of this matrix is done using an accelerometer or other device that can measure roll (ξ) and pitch (ψ). Without certain mechanical restraints, the roll and pitch angles of the accelerometer are not related to the body frame by successive Euler angles. This is called non-orthogonal rotation. If this mechanical restraint was in place, or the matrices were orthogonal, then the Γ matrix would be the product of the roll and pitch rotation matrices,

$$R2(\psi)R1(\xi) = \begin{bmatrix} cos(\psi) & 0 & -sin(\psi) \\ 0 & 1 & 0 \\ sin(\psi) & 0 & cos(\psi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\xi) & sin(\xi) \\ 0 & -sin(\xi) & cos(\xi) \end{bmatrix} \tag{2.17}$$

The Body frame corresponds directly to the optical axis, and the Local frame is corresponding to the position of the entire apparatus. This way, the camera optical axis and rover zenith direction are not required to be aligned. Here, the

two algorithms diverge in methods. The UNH algorithm uses vector methods to determine the shortest arc between two points. The algorithm developed by Texas A&M uses a geometric method to relate the accelerometer readings.

2.3.1 Modified Attitude Matrix Method Γ Matrix

The modified attitude matrix method Γ matrix is computed by the following general steps. The output of the inclinometer (accelerometer) is expressed as a unit vector offset from the rover local nadir vector. The accelerometer output is assumed to have the form of Eq.(2.18)

$$Acc_{measured} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = xi + yj + zk \tag{2.18}$$

The 3x3 Γ matrix is then formulated column by column with the first column, corresponding to the x-axis being derived as follows.

Modified x-axis formulation: Let η_x and θ_1 represent the unnormalized pointing vector about the x-axis. The rotation vector about the x axis is given as:

$$\eta_x^{'} = \mathbf{i} \times (x\mathbf{i} + y\mathbf{j} + z\mathbf{k}) = x\mathbf{k} - z\mathbf{j} \tag{2.19}$$

Where $\times$ symbol represents the vector cross product. It is desired that this will be put into quaternion form, so normalizing the vector is required. This gives

$$\eta_x = \frac{y\mathbf{k} - z\mathbf{j}}{\sqrt{y^2 + z^2}} \tag{2.20}$$

Then, using the vector dot product definition, the angle between the rover nadir vector and the measurement of the gravitational field can be found.

$$\mathbf{A} \cdot \mathbf{B} = (|\mathbf{A}|)(|\mathbf{B}|)cos(\theta) \tag{2.21}$$

Leads to

$$\theta_1 = cos^{-1}[\mathbf{i} \cdot \frac{x\mathbf{i} + y\mathbf{j} + z\mathbf{k}}{\sqrt{x^2 + y^2 + z^2}} = cos^{-1}\frac{x}{\sqrt{x^2 + y^2 + z^2}} \tag{2.22}$$

This projection of the unit gravitational vector can be related to a quaternion, q_1 such that:

$$q_1 = \begin{bmatrix} sin(\frac{\theta_1}{2}) * 0 \\ sin(\frac{\theta_1}{2}) * \frac{-z}{\sqrt{y^2+z^2}} \\ sin(\frac{\theta_1}{2}) * \frac{y}{\sqrt{y^2+z^2}} \\ cos(\frac{\theta_1}{2}) \end{bmatrix} \tag{2.23}$$

Then, converting from quaternions to Euler angles, the first column of Γ can be found as:

$$\Gamma_1 = \begin{bmatrix} x \\ \frac{-y}{\sqrt{y^2+z^2}}\sqrt{1-x^2} \\ \frac{-z}{\sqrt{y^2+z^2}}\sqrt{1-x^2} \end{bmatrix} \tag{2.24}$$

This pattern can be repeated twice more, normalizing against $\mathbf{j} \times (x\mathbf{i}+y\mathbf{j}+z\mathbf{k})$ and taking the dot product of the sensor reading with $\mathbf{j} \cdot (x\mathbf{i}+y\mathbf{j}+z\mathbf{k})$, then again with $\mathbf{k} \times (x\mathbf{i}+y\mathbf{j}+z\mathbf{k})$ and doing the dot product of the reading with $\mathbf{k} \cdot (x\mathbf{i}+y\mathbf{j}+z\mathbf{k})$, to yield the following matrix after using the form $\Gamma_{Total} = [\Gamma_1, \Gamma_2, \Gamma_3]$

$$\Gamma_{Total} = \begin{bmatrix} x & \frac{-x}{\sqrt{x^2+z^2}}\sqrt{1-y^2} & \frac{-x}{\sqrt{x^2+y^2}}\sqrt{1-z^2} \\ \frac{-y}{\sqrt{y^2+z^2}}\sqrt{1-x^2} & y & \frac{-y}{\sqrt{x^2+y^2}}\sqrt{1-z^2} \\ \frac{-z}{\sqrt{y^2+z^2}}\sqrt{1-x^2} & \frac{-z}{\sqrt{x^2+z^2}}\sqrt{1-y^2} & z \end{bmatrix} \tag{2.25}$$

Some properties of this matrix are that it uses three accelerometers arranged orthogonally from each other in order to determine Γ. This means with a predefined ordering of rotation matrices, there are many different ways for this matrix to be implemented, it will be shown in the next section that this task can be completed with a single 3-axis accelerometer. The most important flaw to this matrix is that it was not the one used in testing with SkyScout in Perkins' work. The matrix used in that study is shown in Eq. (2.26). A brief study was performed to determined the equivalence of Γ_{Total} and the matrix used in the SkyScout experimentation

Γ_{EXP}, found in Eq.(2.26)

$$\Gamma_{EXP} = \begin{bmatrix} cos(Altitude) & sin(Altitude)sin(Azimuth) & -sin(Altitude)cos(Azimuth) \\ 0 & cos(Azimuth) & sin(Azimuth) \\ sin(Altitude) & -cos(Altitude)sin(Azimuth) & cos(Altitude)cos(Azimuth) \end{bmatrix} \tag{2.26}$$

Two matrices $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ are equal if they have the same number of rows, the same number of columns, and for each i and j, $a_{ij} = b_{ij}$. Equal matrices have the same dimensions, and objects located in the same positions in the matrices must be equal[22]. This makes proving that $\Gamma_{Total} \neq \Gamma_{EXP}$ simple, since there are mxn chances to show inequality. In this case, the element in the second row, first column of each Γ_{Total} and Γ_{EXP} are examined.

$$\frac{-y}{y^2 + z^2}\sqrt{1 - x^2} = 0 \tag{2.27}$$

Algebraic manipulations can be performed to show that that the conditions of Eq.(2.27) are met when $x = 1$ and when $y = 0$, as long as $y^2 + z^2 \neq 0$. For any other values of x and y, it can be seen that the condition is not met, which is evidence against using Γ_{Total} in experiment.

It is possible to use only a single 3-axis accelerometer to obtain the desired rotation sequence [20]. This is the way that modern smartphones compute their orientation. The theory for this is presented below. The axis definitions are provided in Fig. 2.3. The rotation sequences are defined in Eqs.(2.28),(2.29), and (2.30).

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \tag{2.28}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2.29}$$

Figure 2.3: Axis Definitions for Accelerometer Measurement

$$\mathbf{R}_z = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.30}$$

$$\mathbf{R}_{xyz} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.31}$$

For a 3-2-1 rotation sequence, $\mathbf{R}_{xyz}$, which represents the resulting rotation matrix, is obtained in Eq.(2.32).

$$= \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ cos(\psi)sin(\theta)sin(\psi) - cos(\phi)sin(\psi) & cos(\theta)cos(\psi) + sin(\theta)sin(\phi)sin(\psi) & cos(\theta)cos(\psi) \\ cos(\phi)cos(\psi)sin(\theta) + sin(\theta)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - cos(\psi)sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.32}$$

$$\mathbf{R}_{xyz} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta)sin(\phi) \\ cos(\theta)cos(\phi) \end{bmatrix} \tag{2.33}$$

Given a normalized accelerometer reading $\mathbf{G}_p$:

$$\frac{\mathbf{G}_p}{\|\mathbf{G}_p\|} = \frac{1}{\sqrt{G_{px}^2 + G_{py}^2 + G_{pz}^2}} \begin{bmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta)sin(\phi) \\ cos(\theta)cos(\phi) \end{bmatrix} = \Gamma_{phone} \tag{2.34}$$

Then, solving for the roll and pitch angles, ϕ and θ respectively yields the following solution:

$$tan(\phi_{xyz}) = \frac{G_{py}}{G_{pz}} \tag{2.35}$$

$$tan(\theta_{xyz}) = \frac{-G_{px}}{G_{py}sin(\phi) + G_{pz}cos(\phi)} = \frac{-G_{px}}{\sqrt{G_{py}^2 + G_{pz}^2}} \tag{2.36}$$

2.3.2 Attitude Matrix Method Γ

The attitude matrix formulated method also looked at the misalignment matrix and worried about non-orthogonal rotations. The tilt measured from the accelerometer cannot be related by a straightforward Euler angle rotation. The Attitude Matrix Method instead looked at the tilt from the accelerometer as a roll (ξ), and pitch (ψ). Note that in the previous derivation, these quantities were equal to ϕ and θ respectively.

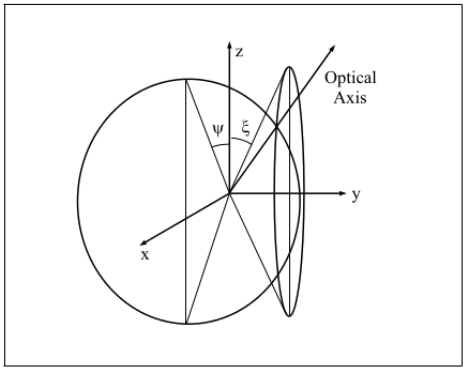

Figure 2.4: Geometric Solution for off axis alignment

Using geometry, this method models the difference between the rover zenith andj the new accelerometer reading as the intersection between two cones. Looking

at the equation of a sphere in $\mathbb{R}^3$.

$$x^2 + y^2 + z^2 = r^2 \tag{2.37}$$

If the roll angle is known, and the pitch angle is varied from 0 to 2π, then the radius of the circle end of the cone is given by

$$r = cos(\psi) \tag{2.38}$$

Thus, the circle lies on the plane of $x = sin(\psi)$. Substitution yields

$$y^2 + z^2 = r^2 = 1 - sin^2(\psi) \tag{2.39}$$

The same line of reasoning could be applied to the case where the pitch angle is known and the roll angle is varied to yield

$$x^2 + z^2 = r^2 = 1 - sin^2(\xi) \tag{2.40}$$

Since both of the points where the cones intersect will satisfy $x = sin(\psi)$ and $y = sin(\xi)$, z can be solved for. Substitution yields

$$z = \pm\sqrt{1 - sin^2(\psi) - sin^2(\xi)} \tag{2.41}$$

This means that the optical axis vector can be defined as follows

$$\vec{oa} = \begin{bmatrix} sin(\psi) & sin(\xi) & \sqrt{1 - sin^2(\psi) - sin^2(\xi)} \end{bmatrix} \tag{2.42}$$

From here, the derivation used the properties of orthogonal vectors to determine the other two orthogonal optical axes. The equations are shown below

$$y \cdot z = 0 \tag{2.43}$$

$$x \cdot z = 0 \tag{2.44}$$

$$y \cdot x = 0 \tag{2.45}$$

$$y \times z = x \tag{2.46}$$

$$x \cdot x = 1 \tag{2.47}$$

$$y \cdot y = 1 \tag{2.48}$$

Then, the generated vector components are normalized and put into final matrix form

$$\Gamma = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} y \times z \\ y \\ z \end{bmatrix} \tag{2.49}$$

2.3.3 Failure Conditions for Zenith Requirement Solution

Eq. (2.42) has a failure condition. As $sin^2(\psi)+sin^2(\xi)$ approaches 1, the third term of the vector enters the complex domain. That is, wherever $sin^2(\psi)+sin^2(\xi) > 1$, the function defining the optical axis z-component no longer has a real value, despite that z may be greater than zero for a given pitch and roll. This suggests that the solution does not produce valid results for useful angles of pitch and roll. Combinations of pitch and roll angles that follow this condition can be seen in Fig. 2.5. It can be seen from the experimental verification of the CST that the orientation angle of the the CST from the vertical never varied by more than approximately 2 degrees. This shows that although CST was not pointed at full zenith, it did not appear to examine a wide range of angles to prove that the zenith requirement could be eliminated. A table depicting the pointing angles of CST is shown

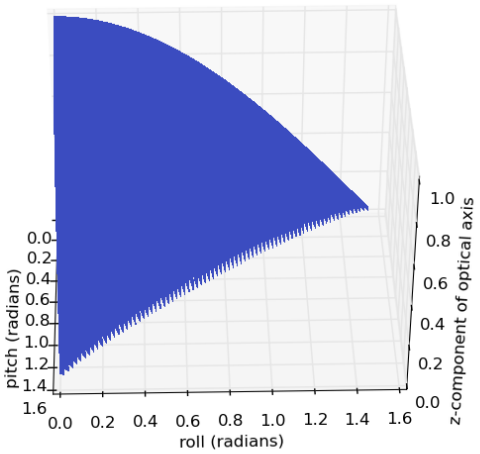

Figure 2.5: Real components of $\sqrt{1 - sin^2(\psi) - sin^2(\xi)}$. The area of the roll-pitch plane where no z axis value is present is an area in which the optical axis z component does not take on a real value.

in Fig 2.6. The inflexibility of the zenith requirement, along with the apparent

Time	ALT	AZM
21:20:30	88d 26m 07s	27d 21m 22s
21:29:30	88d 22m 43s	26d 10m 39s
21:31:50	88d 23m 31s	28d 06m 07s
21:35:00	88d 23m 59s	27d 53m 57s
21:41:00	88d 25m 30s	27d 38m 36s

Figure 2.6: Altitude (Zenith) Angle Orientation of CST

failure condition of the equation that removes the zenith requirement from CST led to the development of a new algorithm, based on maritime techniques.

2.3.4 Miscellaneous Corrections to Attitude Matrix Method

For improved accuracy of the algorithm, and since, for proof of concept, the algorithm will be tested on the Earth, further corrections must be made. model the refraction that the Earth's atmosphere applies to the incoming starlight using Snell's Law[23]. Even though the Earth's atmosphere is a continuum that dissipates as altitude is increased, we model it the atmosphere as a continuous boundary, like when a cup refracts the light of a straw. Snell's Law is depicted in Fig. 2.7 and given as:

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \tag{2.50}$$

where n_1, and n_2 are the indices of refraction for the two mediums, $theta_1$ and $theta_2$ are the angle of incidence, and the angle of refraction, respectively.

For optical axis angels that are less than 45° from zenith, the angle of refraction $\theta = \theta_1 - \theta_2$ can be approximated as

$$\theta = (n_2 - 1) \tan(\theta_2) \tag{2.51}$$

The angular correction for angles from aligned with zenith (0 rads) to 45° is shown

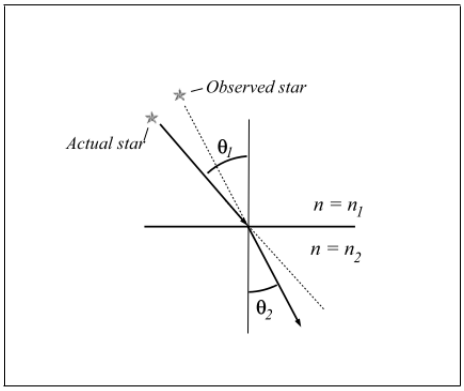

Figure 2.7: Snell's Law and Atmospheric Aberration Effect

in Fig. 2.8. The other major matrix that is constructed as part of the Attitude

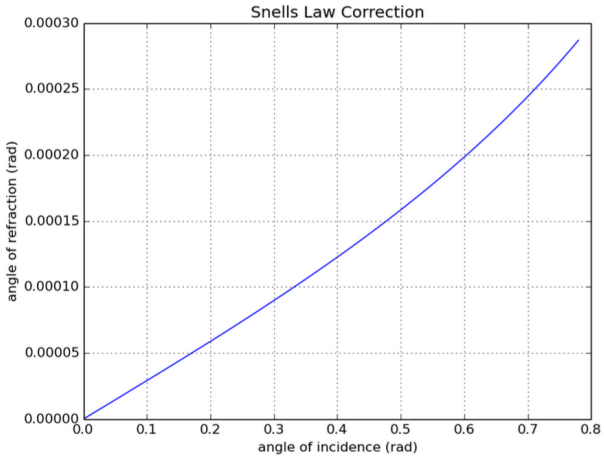

Figure 2.8: Plot of Angle of Incidence vs. Angle of Refraction for Snell's Law

Matrix Method is the $A^{G/I}$ matrix. These corrections are important because it is possible to model the irregular swaying of the Earth's rotational rotational axis. Fabricating this matrix is challenging because the translation between Greenwich to Inertial coordinate frames depends on many physical factors that are not easy to predict. The main components for this matrix are the nutation and precession. The other secondary components are the Greenwich angle ψ and the Earth orientation parameters, which are also referred to as the polar motion. Fig. 2.9 shows what nutation and precession look like with respect to a planetary body's rotational axis.

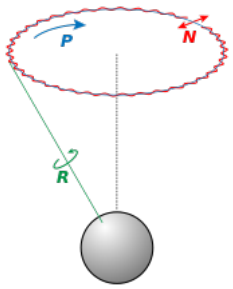

Figure 2.9: Nutation and Precession

Here, P shows the precession, or the minute circular motion the Earth tends

to take over a long period of time, and N shows the nutation, or the oscillations within that circular path. Vallado created a curve fitting method that allows the nutation and precession of the Earth to be modeled as follows[24].

$$NP = R_1(-\alpha)R_3(-\beta)R_1(\gamma)R_3(\delta) \tag{2.52}$$

γ and δ are define the location of the ecliptic pole in the inertial frame, β is the ecliptic angle of precession, and α is the obliquity of the ecliptic. These angles are all functions of time and are defined as follows.

$$\alpha = (84381.4428 - 46.8388t - 0.0002t^2 + 0.002t^3)/3600 \tag{2.53}$$

$$\beta = (-0.0431 + 5038.4739t + 1.5584t^2 - 0.0002t^3)/3600 \tag{2.54}$$

$$\gamma = (84381.4479 - 46.814t + 0.0511t^2 + 0.0005t^3)/3600 \tag{2.55}$$

$$\delta = (10.5525t + 0.4932t^2 - 0.0003t^3)/3600 \tag{2.56}$$

In these equations, time is calculated as a function of the Julian Date, as given in the following equation

$$t = \frac{JD - T_0}{T_{century}} \tag{2.57}$$

here, JD is the Julian Date, T_0 is the Julian Date at J2000 ($T_0 = 2451545$), at $T_{century}$ is equal to 36525, or the number of days in one century.

2.4 Proposed Algorithm

2.4.1 Blind Astrometric Nautical Sight Reduction

The proposed method is a computer vision based method that combines the Blind Astrometric Calibration Method, the Center Pixel Method, and the Direct Com-

putation Method for Nautical Sight Reduction [16]. This method measures the angle between the horizon and star. This angle is known as the altitude. The algorithm requires an optical light image of the stars. The raw image from Fig. 2.11 processed by the astrometry software. An example result of "image 60", used for actual position determination, is shown in Fig. 2.12. The astrometry package is a complicated piece of software, which uses image processing, geometric methods, and access to the USNO-B star catalog for determination of the celestial bodies in the image[25]. A high level overview of the Blind Astrometric Nautical Sight Reduction Algorithm is shown below in Fig. 2.10.

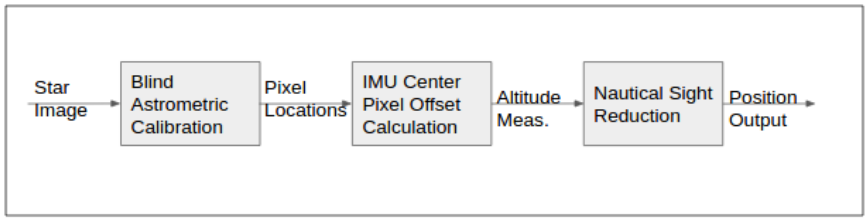

Figure 2.10: Top Level Block Diagram of New Algorithm

Figure 2.11: Example image taken by star tracker.

The algorithm begins with an estimation of the initial position of the mobility

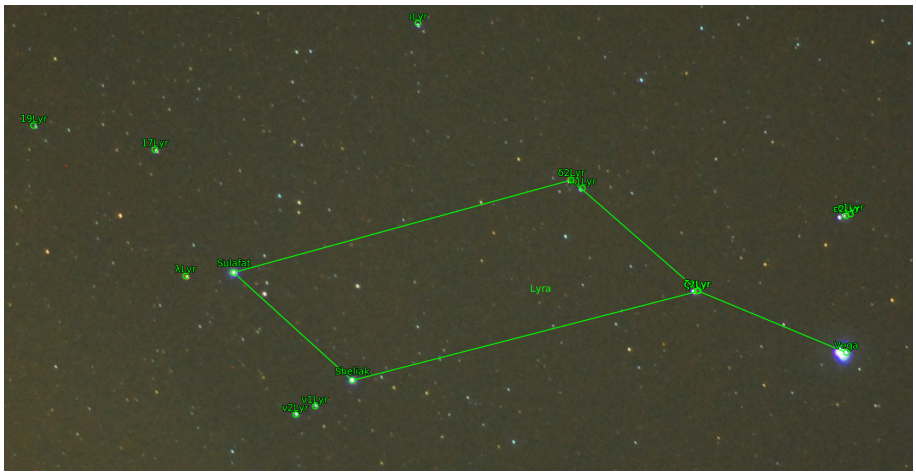

Figure 2.12: Example image after processing by Astrometry software package.

solution. The local hour angle is computed as

$$LHA = GHA + Long \tag{2.58}$$

H_c is the altitude of the observed star if the estimated position was the true location. H_c can be calculated as follows.

$$H_c = \sin^{-1}(S\sin(Lat) - C\cos(Lat)) \tag{2.59}$$

In equation (2.59), S is defined as $S = \sin(Dec)$ and C is defined as $C = \cos(Dec)\cos(LHA)$. X, and A are parameters which are used to determine the azimuth direction, and are calculated as follows.

$$X = \frac{S\cos(Lat) - C\sin(Lat)}{\cos(H_c)} \tag{2.60}$$

$$A = \cos^{-1}(X) \tag{2.61}$$

In Eq. (2.61), if $LHA > 180°$, then set Z = A. Otherwise, $Z = 360° - A$. The next portion of the sight reduction method is an iterative method which calculates

ground distance between the estimated point from H_c and the observed point found with H_o, known as p. H_o is the measured altitude of the star. The quantity p is also referred to as the intercept Eq. (2.62). In this research, the quantity is also referred to interchangeably as ΔH.

$$p = H_o - H_c \tag{2.62}$$

Fig. 2.13 depicts the Nautical Sight Reduction method, and in particular Eq.(2.62).

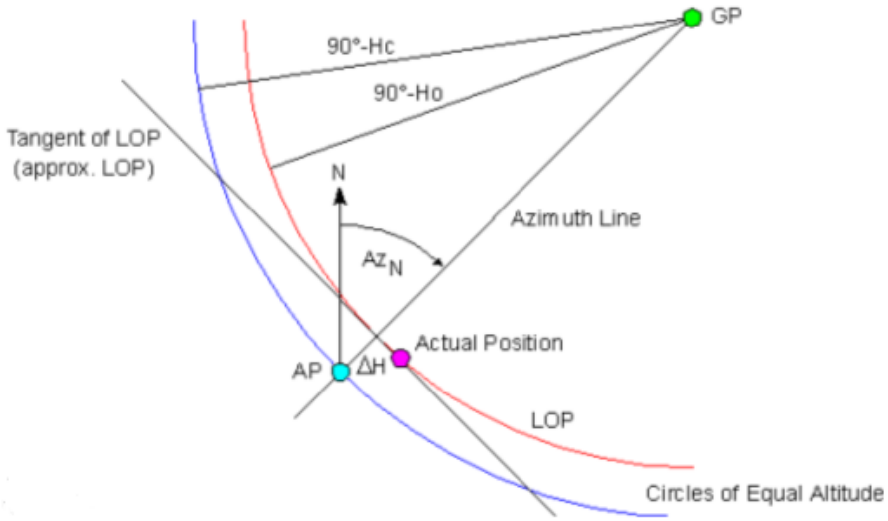

Figure 2.13: Diagram of Sight Reduction. Note ΔH on this figure is defined as p in this research [15]

2.4.2 Center Pixel Method

Calculation of H_o is performed by assigning IMU pitch and roll values to the image center pixel. This is shown in Eq. (2.63) where the variables are in units of pixels.

$$\begin{pmatrix} x_T \\ y_T \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y_c + y_{off} \end{pmatrix} \tag{2.63}$$

Now that Z and p for each observation are calculated, the terms from the iterative method can be calculated. The terms, A,B,C,D,E and G of each summation are used to calculate the center of the error polygon used for navigation. Note that n represents the number of observations. The terms are defined in as follows:

$$A = \sum_{i=1}^{n} \cos^2 Z_i \tag{2.64}$$

$$B = \sum_{i=1}^{n} \cos Z_i \sin Z_i \tag{2.65}$$

$$C = \sum_{i=1}^{n} \sin^2 Z_i \tag{2.66}$$

$$D = \sum_{i=1}^{n} p_i cos Z_i \tag{2.67}$$

$$E = \sum_{i=1}^{n} p_i sin Z_i \tag{2.68}$$

These terms can then be combined as follows. Note B_I and B_F is the improved and estimated latitude respectively. L_I and L_F is the improved and estimated longitude respectively. d is the distance in nautical miles between (L_F, B_F) and (L_I, B_I). This value is expected to converge as the number of iterations increase, indicating that a final position estimate is obtained.

$$G = AC - B^2 \tag{2.69}$$

$$B_i = B_F + \frac{CD - BE}{G} \tag{2.70}$$

$$L_i = L_f + \frac{AE - BD}{G cos(B_F)} \tag{2.71}$$

$$d = 60\sqrt{(L_I - L_F)^2 cos^2(B_F) + (B_I - B_F)^2} \tag{2.72}$$

The algorithm is repeated until the value of Eq. (2.72) is sufficiently small. The process is depicted in the convenient diagram form of Fig. 2.14. It is expected that

the zenith requirement and backtracker method will be eliminated by the correct implementation of this algorithm.

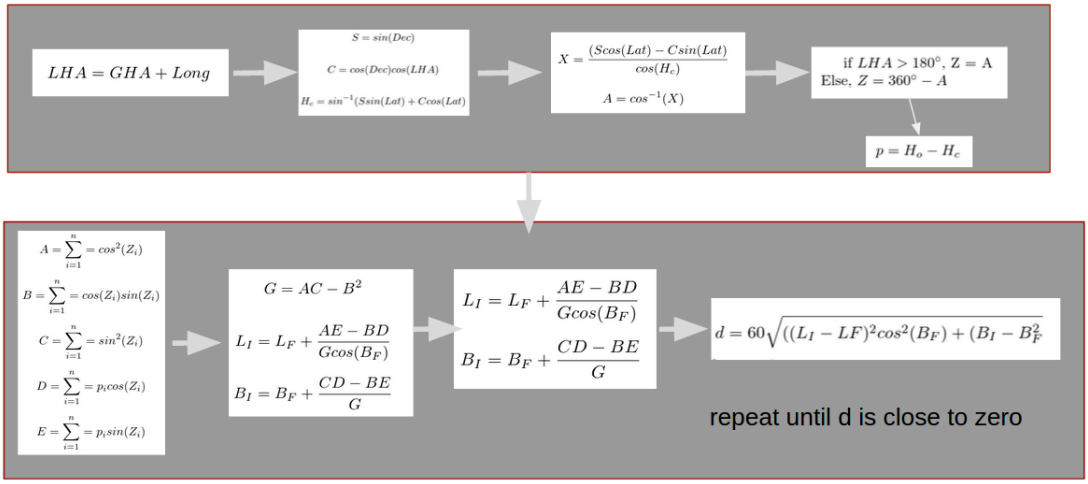

Figure 2.14: Diagram Form of Nautical Sight Reduction Algorithm

Chapter 3

Simulations of Nautical Sight Reduction Algorithm

Simulations were performed to demonstrate the use of the nautical sight reduction method. These simulations also show convergence and accuracy of three different observation formats. **Case 1** sights three stars sighted at three different times. **Case 2** sights three stars from the same image. **Case 3** sights a single star from three different images, taken at three different times. Tables 3.1 and 3.2 depict the right ascension and declination of Case 1 and Case 2, repsectively. Table 3.3 shows the right ascension and declination of the summer triangle. This includes the data on the star Altair, which comprises Case 3. The values of right ascension and declination were obtained from the Hipparcos star catalog [26].

.	Regulus	Antares	Kochab
Right Ascension	10h08m22s	16h29m25s	14h50m25s
Declination	$11°58'$	$-26°25'$	$74°09'$

Table 3.1: Right Ascension and Declination of Stars Observed in Case 1

.	Sulafat	Sheliak	Vega
Right Ascension	18h58m56s	18h50m04s	18h36m56s
Declination	$32°41'$	$33°21'$	$38°47'$

Table 3.2: Right Ascension and Declination of Stars Observed in Case 2

.	Altair	Deneb	Vega
Right Ascension	19h50m46s	20h41m26s	18h36m56s
Declination	$08°52'$	$45°16'$	$38°47'$

Table 3.3: Right Ascension and Declination of Stars Observed in Case 3 and Summer Triangle

The stars of the summer triangle were sighted concurrently from Durham, NH, ($70.9349°W$, $43.1338°N$). This test was used in a simulation of algorithm convergence only. In Case 1, the stars were sighted from $15°E$, $32°N$. For this format, the time intervals between sightings are 6 and 25 minutes. Simulations of Case 2 and Case 3 are sighted from Durham, NH ($70.9349°W$, $43.1338°N$). In Case 3, each image of Altair was simulated to be approximately 8 minutes apart.

3.1 Convergence

The first test of each observation format is to determine whether or not Eq. (2.72) converges to a single value. The algorithm was tested over ten iterations with no additional noise or bias. In Figure 3.1 it can be seen that in general each observational format converges by the second iteration. Case 3 takes four iterations to converge. This suggests that method may be suboptimal for navigation.

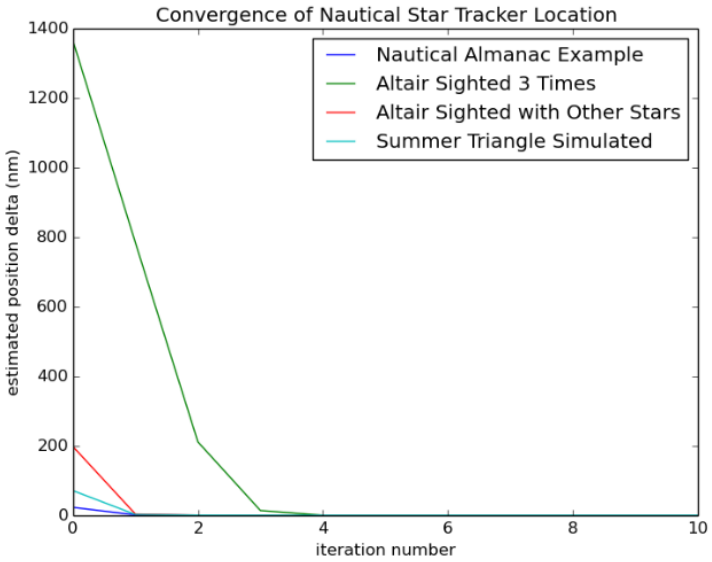

Figure 3.1: Convergence of the different observational schemes.

3.2 Position Estimation

Each of the formats was examined for accuracy of position estimation. The simulation was run 1000 times, and iterated over Eq.(2.72) 10 times. The desired result is to determine the correct altitude measurements for each of the three cases, from which correct position can be obtained. No noise or bias is added to the altitudes in the position estimation simulations. The results and corresponding histograms are presented. Each histogram has a total of 1000 observations, each having the same value. Table 3.4 depicts the legend information for plots of this section.

Blue	Red	Green	Teal
Position Est.	N/A	N/A	True Value

Table 3.4: Legend Format for Position Estimate Plots

Case 1: Three Stars, Three Times

The histogram of position estimates corresponding to the stars Kochab, Regulus and Antares observed from $15°E$, $32°N$ at different times from are shown in Fig. 3.2. The map of these estimates is shown in Fig. 3.3. These results suggest that observing three stars a single time is a potentially viable method for position determination.

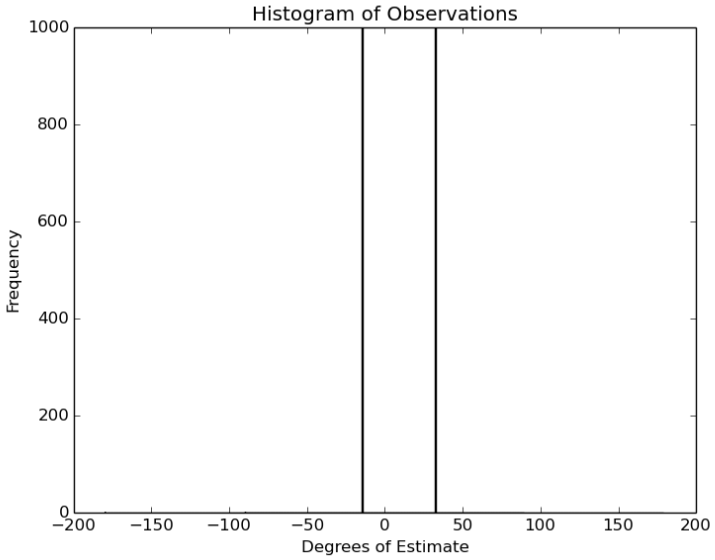

Figure 3.2: Case 1 Histogram of Position Estimates

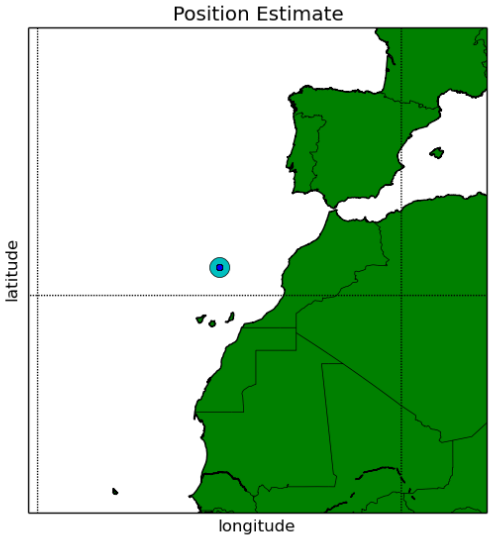

Figure 3.3: Case 1 Map of Position Estimates

Case 2: Three Stars, One Time

The histogram of position estimates corresponding to the stars Sheliak, Sulafat, and Vega observed at the same time, (UTC-01:09:00) is shown in Fig. 3.4. The map of these estimates is shown in Fig. 3.5. These results suggest that observing three stars a single time is a potentially viable method for position determination.

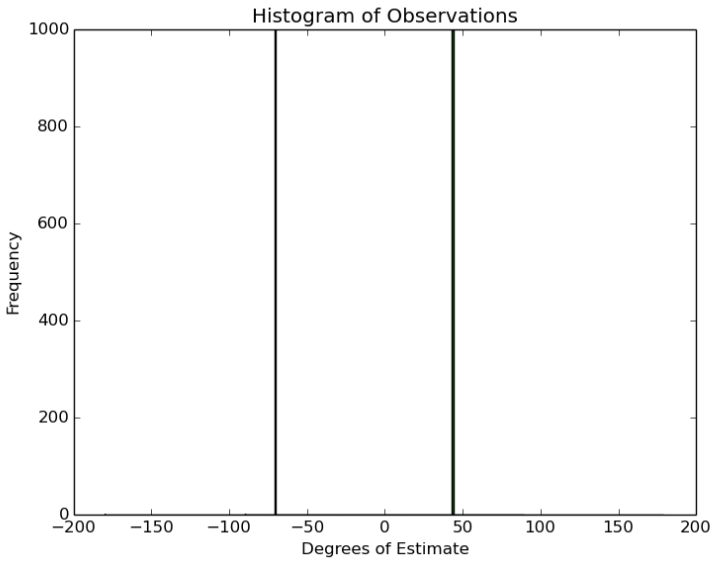

Figure 3.4: Case 2 Histogram of position estimates.

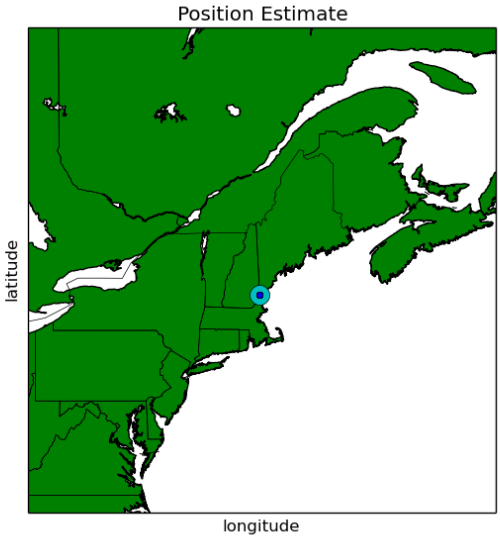

Figure 3.5: Case 2 Map of Position Estimates.

Case 3: One Star, Three Times

The histogram of position estimates corresponding to the star Altair observed three different times is shown in Fig. 3.6. The corresponding map is shown in Fig. 3.7.

These results suggest that observing a single star one time is a potentially viable method for position determination.

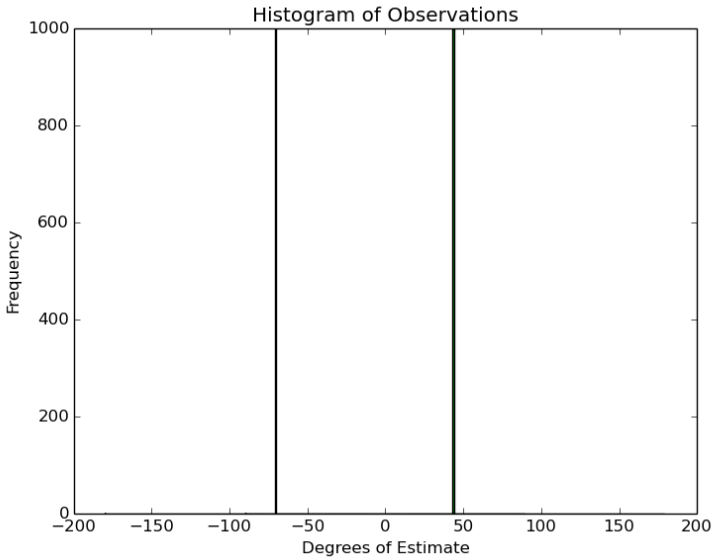

Figure 3.6: Case 3 Histogram of Position Estimates

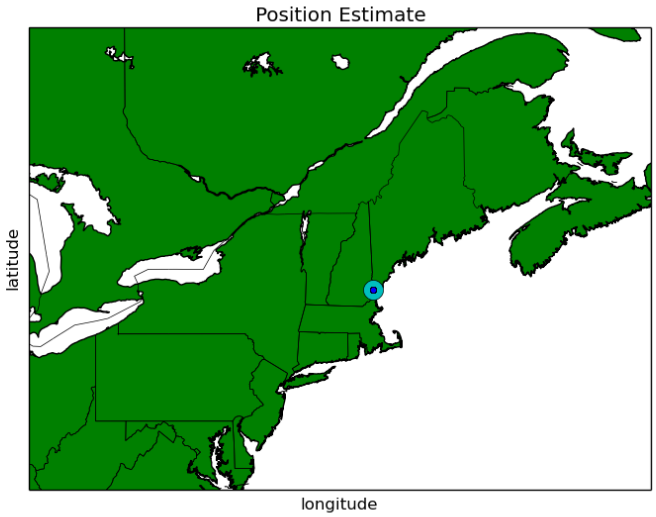

Figure 3.7: Case 3 Map of Position Estimates

3.3 Noise in Observed Altitudes

A uniform distribution of noise was added to each observation to determine the effect of random noise on position estimate. The uniform distribution was chosen because it enabled easy to implement noise bounds while retaining the mean an accurate measure of the distribution center. Additionally, this distribution imposes a more strict requirement on the star camera noise bounds so that a greater accuracy may eventually be obtained. The probability density function of this distribution is described in Eq. (3.1). A plot of a histogram developed using this distribution is shown in Fig. 3.8.

$$p(x) = \frac{1}{b-a} \tag{3.1}$$

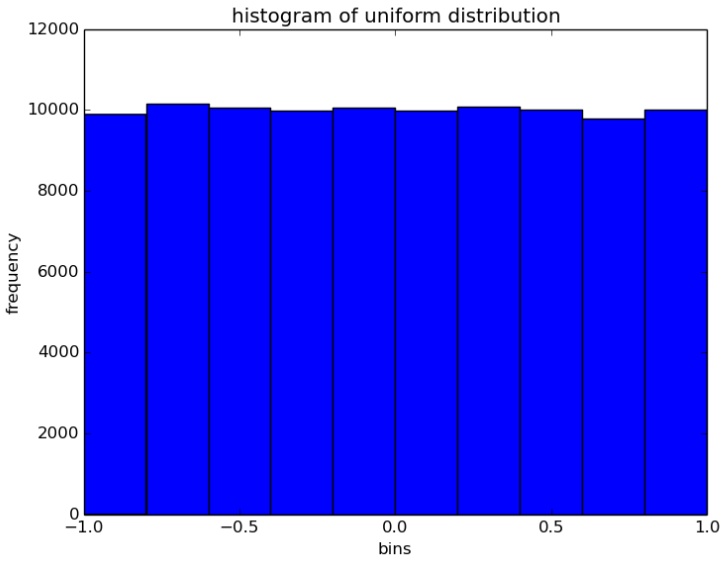

Figure 3.8: Uniform Probability Density Histogram

Table 3.5 depicts the legend of the plots for this section.

Blue	Red	Green	Teal
$\pm 1°$	$\pm 2°$	$\pm 5°$	True Value

Table 3.5: Legend of Noisy Position Estimate Plots

Studies of the noise examine the geometric distribution of the estimated points and determine if the mean of the estimates yields the correct position. For each data point generated in the study, a simulated array of three altitude measurements, $H_{o,1}$, $H_{o,2}$, and $H_{o,3}$ are generated. Two cases were studied. The boundaries (a, b) for the noise ϵ_i were chosen to be ± 1,± 2, and ± 5 degrees. In the uniform case, the same amount of noise is added to each star altitude measurement such that, for an additional noise value ϵ_i, the altitudes are input to Nautical Sight

Reduction as depicted in Eq. (3.2).

$$\begin{bmatrix} H_{o,1,new} \\ H_{o,2,new} \\ H_{o,3,new} \end{bmatrix} = \begin{bmatrix} H_{o,1} + \epsilon_1 \\ H_{o,2} + \epsilon_1 \\ H_{o,3} + \epsilon_1 \end{bmatrix} \tag{3.2}$$

The expected effect of the uniform simulation of uniform noise is that the values of noise will function similar to a value of bias. If the same constant value ϵ_1 is added to each $H_{o,i}$ of an observation, the circles of equal altitude will increase for $-\epsilon_1$ and decrease for $+\epsilon_1$. In the non-uniform case, a different amount of noise is added to each observation, described in Eq. (3.3).

$$\begin{bmatrix} H_{o,1,new} \\ H_{o,2,new} \\ H_{o,3,new} \end{bmatrix} = \begin{bmatrix} H_{o,1} + \epsilon_1 \\ H_{o,2} + \epsilon_2 \\ H_{o,3} + \epsilon_3 \end{bmatrix} \tag{3.3}$$

The expected effect of a non-uniform noise is that the distribution of points will take up more area. This is because position estimate determined by Eq. (2.70), and Eq. (2.71) is determined by taking the centroid of the spherical error polygon. As the radius one of the circles of position change due to the value of ϵ_i, the estimated position will also change.

Case 1: Three Stars, Three Times

Fig. 3.9 and Fig. 3.10 depict the uniform and non-uniform noise on the observations of Kochab, Antares, and Regulus.

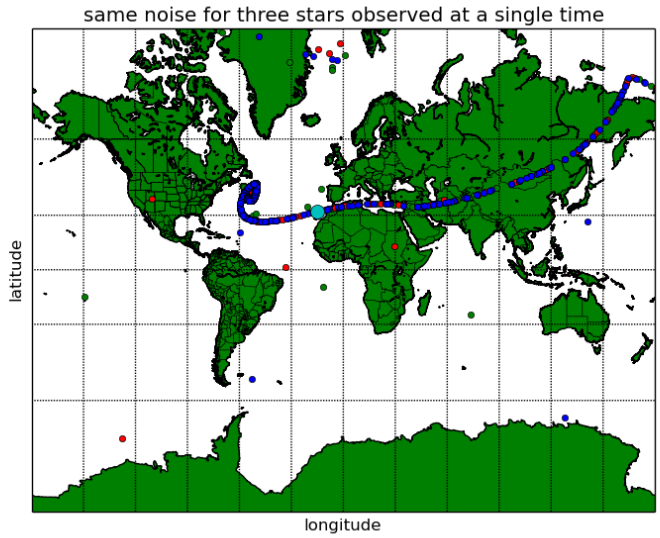

Figure 3.9: Distribution of Estimates for Case 1 Observation Format: Uniform Noise Simulation

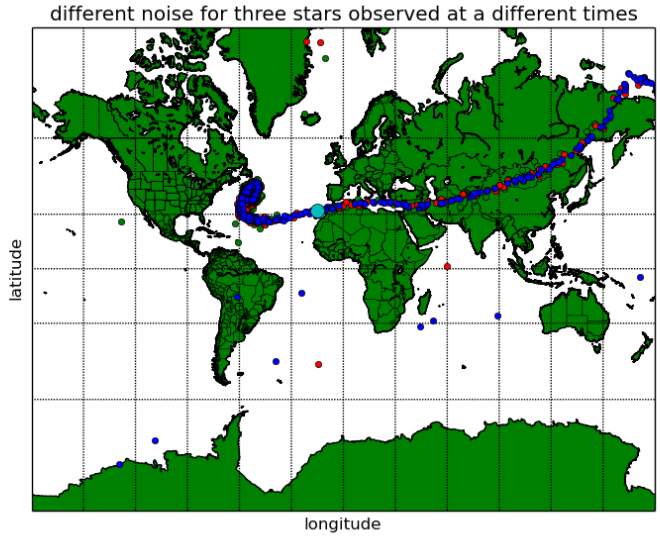

Figure 3.10: Distribution of Estimates for Case 1 Observation Format: Non-Uniform Noise Simulation

One of the most notable effects of this format, especially as compared to the other formats studied of this chapter are that the distribution of the estimates has the same shape for both the uniform and non-uniform noise simulations. One reason for the interesting curvature may be that the stars are in different parts of the sky compared to the stars of the other two observational formats. The right ascensions and declinations of Regulus, Antares, and Kochab are given in Table 3.1. This format was observed to have a large number of non-convergent values. If the lines of position, as calculated from Eq. (2.62) and Eq. (2.61) do not intersect, then a position estimate is not able to be obtained using this observational format. The presence of such nonconvergent values infers that Case 1 is not appropriate for practical use until further study can be performed.

Case 2: Three Stars, One Time

Fig. 3.11 depicts the estimated position given the same amount of uniform random noise added to each of the three stars observed. Fig. 3.12 depicts the estimated positions given different amounts of uniform random noise for each star. It can be observed that a uniform noise offsets the position estimate in a linear fashion. The centroid of the distribution of estimates represents the true point.

In Fig. 3.12, it can be observed that for noise bounds of $\pm 1^\circ$, the distribution of points approximates a straight line. As noise increases, as seen in the set of points corresponding to $\pm 2^\circ$ and $\pm 5^\circ$ the curvature in the distribution increases, and error in the longitude is dragged to one side. One possible cause is that because the stars are so close together in the sky, the circles of position that they generate to estimate position are very close together. For large values of noise, the estimates trace an approximately circular path defined by these observations. It should also be noted that nonconvergent values were not as common in Case 2, only frequently occurring at noise bounds of $\pm 5^\circ$. This analysis indicates that Case 2 may is a viable candidate for the experimental device..

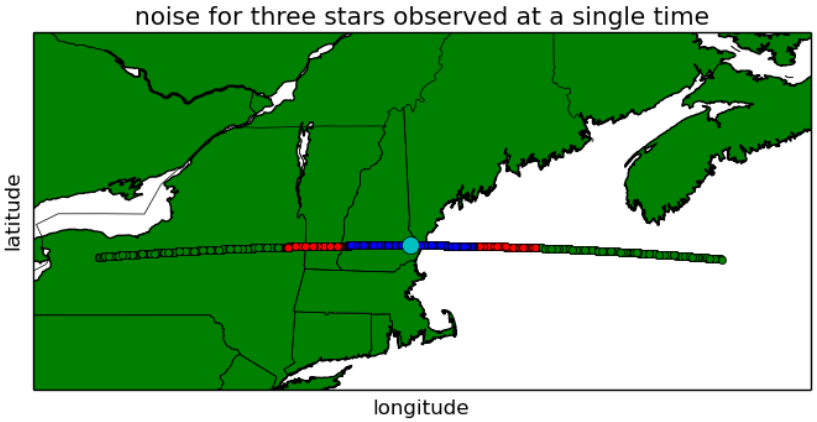

Figure 3.11: Distribution of Estimates for Case 2 Observation Format: Uniform Noise Simulation

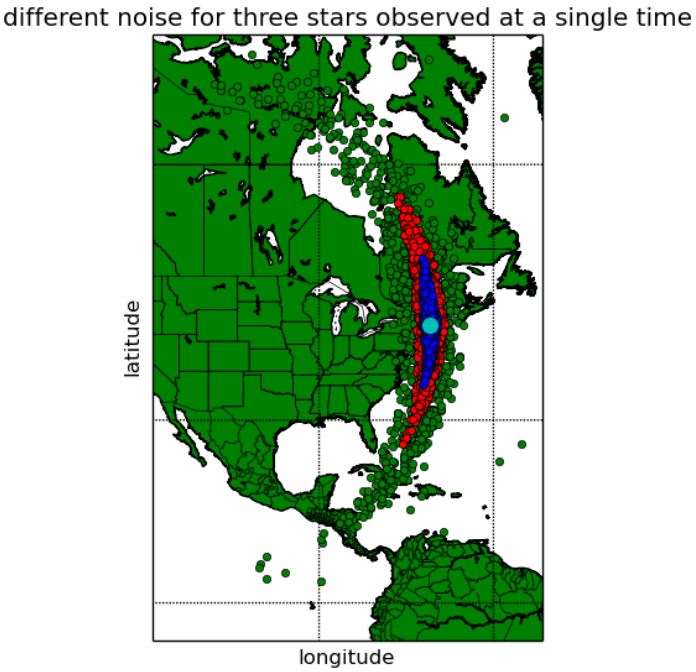

Figure 3.12: Distribution of Estimates for Case 2 Observation Format: Non-Uniform Noise Simulation

Case 3: One Star, Three Times

Fig. 3.13 depicts the Case 3 simulated observation with uniform noise. Fig. 3.14

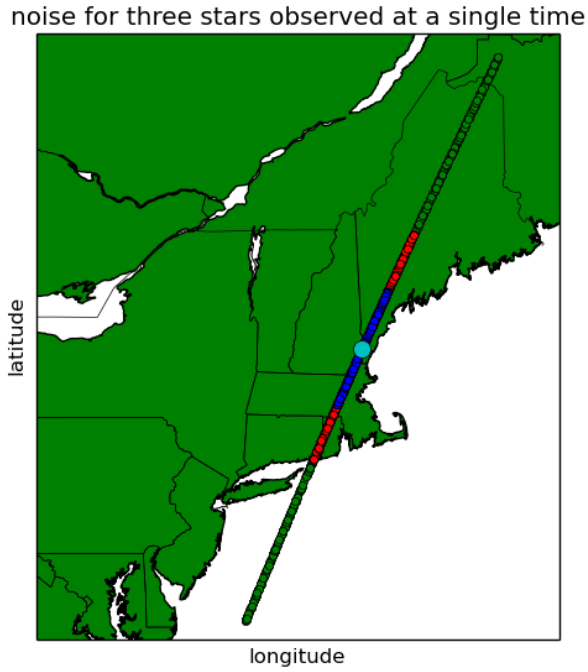

Figure 3.13: Distribution of Estimates for Case 3 Observation Format: Uniform Noise Simulation

depicts the simulated observation of Case three with non-uniform noise. Several values were not plotted because the predicted latitude and longitude were above the range of $(-180, 180)$ for the longitude and $(-90, 90)$ for the latitude. This meant that mean value was not an accurate predictor of location when one star was observed three times. In addition, for points where the randomly generated bias in H_o was high, the algorithm sometimes produced non-convergent results. Case 3 is more sensitive to noise because it is only viewing one star, and the need for an accurate measurement is more pronounced. For this observation format, the mean was not observed to be an accurate predictor of location. Based on the

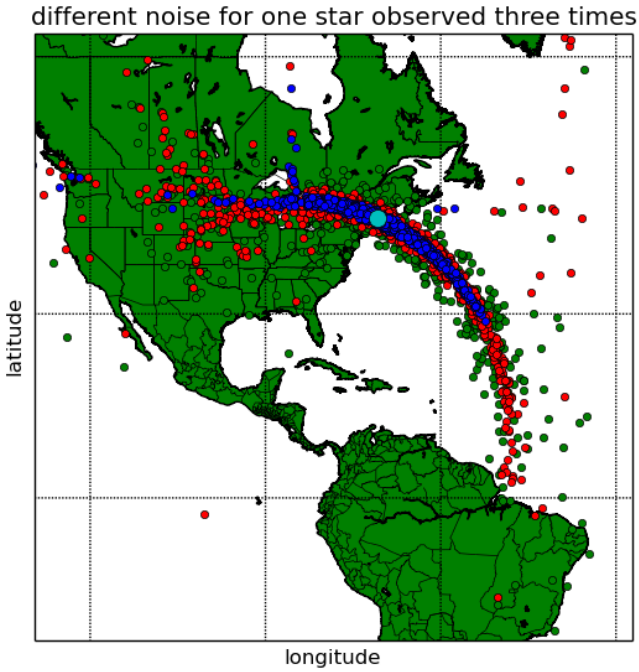

Figure 3.14: Distribution of Estimates for Case 3 Observation Format: Non-Uniform Noise Simulation

above analysis, Case 3 was not selected for use on the star tracker.

3.4 Observation Bias

Bias was artificially introduced into simulations to determine the effects of pitch offset. Two cases were simulated. In the first case, a uniform pitch offset (bias), b_i is added to each of the three noisy star observations. In the second case, different biases are added to each of the three star observations. The key difference between this test and the previous noise tests, is that the values of bias added to the array of $H_{o,1}$, $H_{o,2}$, and $H_{o,3}$ are constant throughout each iteration of the simulation, and are not determined by a probability density function. The array of observed altitudes describing the uniform bias test is shown in Eq. (3.4).

$$\begin{bmatrix} H_{o,1,new} \\ H_{o,2,new} \\ H_{o,3,new} \end{bmatrix} = \begin{bmatrix} H_{o,1} + \epsilon_1 + b_1 \\ H_{o,2} + \epsilon_2 + b_1 \\ H_{o,3} + \epsilon_3 + b_1 \end{bmatrix} \tag{3.4}$$

The array of observed altitude describing the non-uniform bias test is shown in Eq. (3.5).

$$\begin{bmatrix} H_{o,1,new} \\ H_{o,2,new} \\ H_{o,3,new} \end{bmatrix} = \begin{bmatrix} H_{o,1} + \epsilon_1 + b_1 \\ H_{o,2} + \epsilon_1 + b_2 \\ H_{o,3} + \epsilon_1 + b_3 \end{bmatrix} \tag{3.5}$$

Uniform Bias

Fig. 3.15 shows the distribution of uniformly biased, noisy observations from the Case 2 format. b_1 was set equal to 1°, 2° and 5° in this simulation.

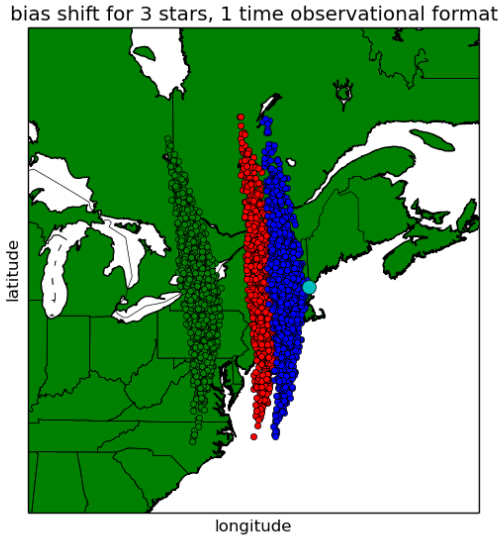

Figure 3.15: Uniform Bias Shift of Position Estimate

Table 3.6 shows the legend information for the Fig. 3.15 simulations.

Blue	Red	Green	Teal
$b_1 = 1°$	$b_1 = 2°$	$b_1 = 5°$	True Value

Table 3.6: Legend Format for Position Estimate Plots

Since the stars of Case 2 are very close together in the sky, and position determination is found by the intersection of three circles of equal altitude, it is currently believed that the distributions of points in currently follow circles of equal altitude. The direction of translation for the distribution may depend on the direction the stars were observed from. Adding uniform bias to each $H_{o,i}$ in the observation vector is expected to the distribution of position estimates based on the orientation of the star camera. Adding bias shifts the estimates towards the ground position of the star. Subtracting bias moves the estimates away from the ground position of the star. This is all observed in the figure and is consistent with comments from Blewitt regarding the size and behaviour of circles of equal

altitude[27]. The mean points for the biased position estimates were examined for linear correlation to learn more about how the circles of positions change. The estimated longitude is plotted against bias in Fig. 3.16. The bias is observed to have a nearly linear response, with $R^2 = 0.99996$. The estimated latitude is plot-

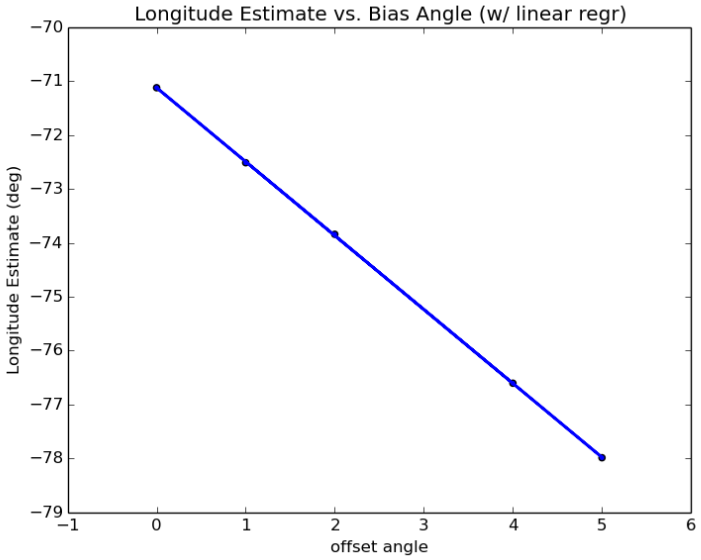

Figure 3.16: Longitude Estimate vs. Bias Shift

ted against bias in Fig. 3.17. In this case the linear model does not fit as well. The correlation coefficient was observed as $R^2 = 0.40295$.

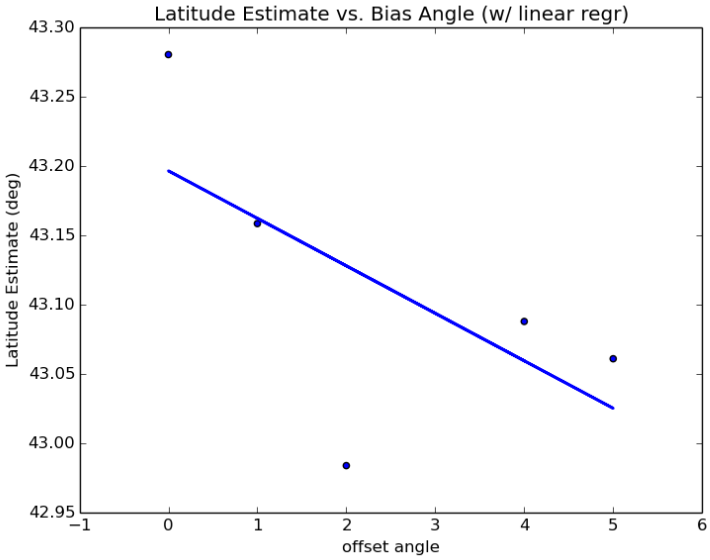

Figure 3.17: Latitude Estimate vs. Bias Shift

Non-Uniform Bias

A short study was performed on the observations of Sulafat, Sheliak, and Vega to examine the effects of non-uniform bias. Uniform noise of $\pm 1^\circ$ was added to each $H_{o,i}$ of the observation vector. A bias of $+1^\circ$ was added to the observation of Sulafat. $+2^\circ$ was added to the observation of Sheliak. $+5^\circ$ was added to the observation of Vega. The position estimates are depicted in Fig. 3.18.

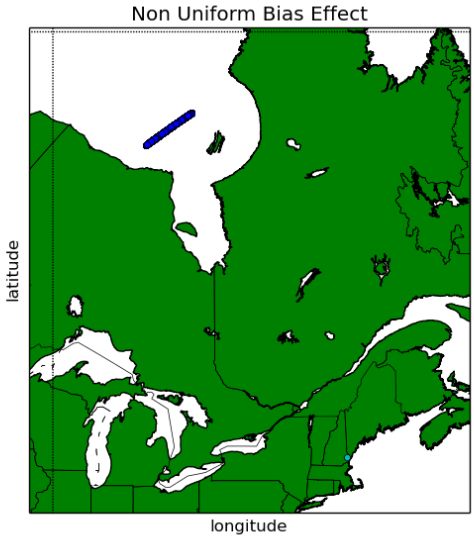

Figure 3.18: Bias Study With Non-Uniform Bias Added to Sheliak, Sulafat, and Vega

The estimated positions are observed to follow a line that has a different trajectory from those observed in the case of simple uniform noise injection. The position estimate is also less accurate.

3.5 Summary of Observational Formats

In summary, Case 2 was the best method simulated in order to obtain a viable position estimate. The mean position estimate is a good indicator of the true position for uniform noise and low values of non-uniform noise. The addition of uniform bias had a measurable effect on the dataset. Case 2 requires only a single image to obtain a position estimate. In addition to having the slowest number of iterations to convergence, Case 3 takes the longest to complete a single observation, since the camera shutter would need to be open for three times as long, and the orientation of the camera may have to change between images.

These difficulties of implementation makes for a suboptimal method of dynamic navigation. Additionally, Case 3 has very little to no robustness to noise, or bias. This means that its application requires better sensors to function properly. Since the geometric distribution of position estimates is non-uniform, the mean is not a good indicator of the position. As such this method was not selected. Case 3 also requires three images to obtain a position estimate. Simulations of Case 1 led to an unexpected distribution of points. This unique distribution is likely caused by the varied locations of the observed stars in the night sky. Additionally, Case 1 requires three images to obtain a position estimate. It may be viable for position determination after further study but ultimately was not selected for use in the experimental device. Overall, each method is able to perform correct position determination given perfect sensors. Based on the analysis of this section, the Case 2 observation format was selected for the experimental prototype.

Chapter 4

Experimental Design and Verification

This section details the hardware, software, and sources of error for the Celestial Navigation Device. Different forms of error are disucssed. Additionally, the experimental results are discussed, along with a generic calibration method which attempts to reduce the amount of error the device exhibits.

4.1 Experimental Setup

The experimental device is a Nikon D60 DSLR camera, with an attached Nikon 50mm f/1.8D Auto Focus Nikkor Lens. The D60 was chosen due to its charge coupled device (CCD) image sensor. The CCD was believed to be important due to its linear response to light compared to a complimentary metal oxide semiconductor sensor (CMOS), which is the other leading type of image sensor found in digital cameras. This factor was considered in the event that the image processing software would be manufactured in house. It was later found that the response of CMOS and CCD sensors are both linear up to a saturation point[28]. This supports that CMOS cameras are also compatible with the star tracker design

presented in this research. The lens is set to the "∞" focus setting. The camera exposure time was eight seconds for each image, and the ISO sensor sensitivity was set to 800. These settings were determined by trial and error to obtain a high quality image. A standard photography tripod is used to stabilize the camera housing along with a remote shutter to prevent movement from the user when the photograph of the stars is taken. To ensure that adequate light is captured for star identification, the shutter remains open for 8 seconds per photograph. The camera flash attachment houses a Razor 9DOF inertial measurement unit (IMU). The IMU reports the Euler angles between the local coordinate frame and the camera optical axis frame. The IMU sets the Euler angles for the center pixel of the camera image. Since useful navigational stars appear in locations other than the center pixel, an offset from the center pixel is applied using the field of view of the camera lens. The camera setup has a field of view across the sensor diagonal of approximately 30.5 degrees. This corresponds to an angular offset of approximately 24.2 arcseconds per pixel away from the center pixel. An image of the experimental setup can be seen below. These values for the field of view per pixel are calculated automatically in the Astrometry.net package and can also be calculated from the Pythagorean theorem. The diagonal field of view of the D60 lens is 46°. The image is 3872Wx2592H pixels. Trigonometry can be used to yield a factor of 24.1 arcseconds per pixel. This fact is further confirmed by the Astrometry package. A laptop is used to record the local time and the IMU Euler angle output. The setup is pictured in Fig. 4.1.

4.1.1 IMU Time Response Curves

It was noted that while under USB configuration, the Razor 9-DOF IMU had a transient response. This means that the pitch angle would changes over time after an initial warm up period. It is presently believed that this "warm-up" time is a property of the IMU. To avoid erroneous position estimation that is associated with the IMU transient, the IMU is powered on for several minutes

Figure 4.1: Experimental setup for the Celestial Navigation Device. The remote shutter can be seen on top of the laptop.

before measurements are taken. A plot of the IMU pitch angle is shown in Fig. 4.2 for several different initial conditions. An interesting revelation of Fig. 4.2 is that

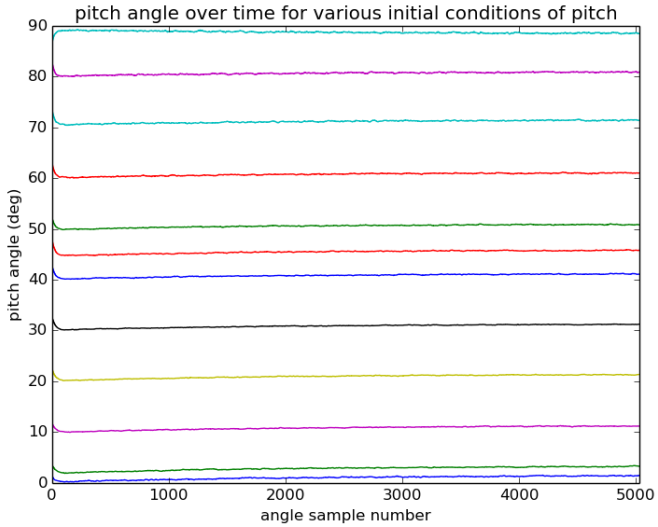

Figure 4.2: Time response for various pitch angles.

the 90 degrees trial has a first order response that is opposite in sign to the other nominal angle trials. Computationally timing the data collection algorithm in a script estimated the sample rate to be 91.2Hz. Two methods of processing are

applied to the dataset according to the block diagram of Fig. 4.3. The first is a

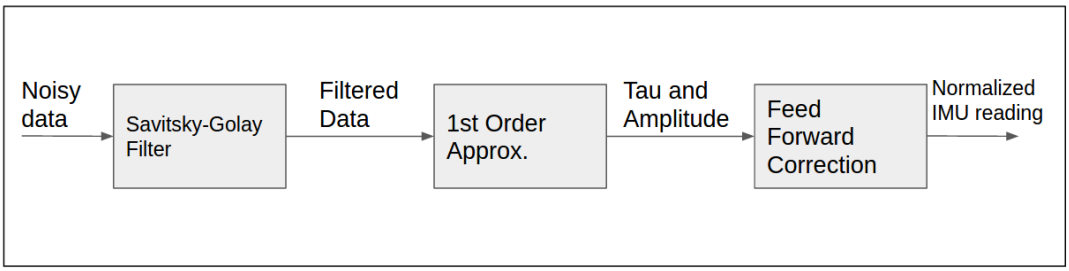

Figure 4.3: Block Diagram for Filtering IMU data

method known as Savitzky-Golay filtering. This filtering mechanism is a digital filter which attempts to increase the signal to noise ratio by uses a moving window and fits different subsets of data to a different polynomial [30]. Fig. 4.4 shows the results of Savitsky-Golay filtering. Note that the angle is drifting to approximately 11 degrees, and that the vertical axis is expressed in scientific notation. The filtered data is the green data.

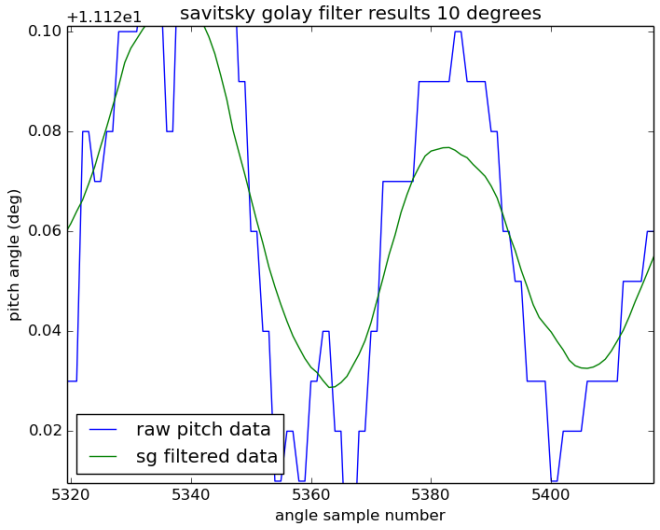

Figure 4.4: Savistky-Golay Filtering.

It should be noted that while this filter does model the data more accurately,

the pitch angle did not move for this experiment. It is possible that the Savitsky-Golay filter may be attempting to model the noise. For systems where a polynomial does represent the signal, Savitsky-Golay filtering may be optimal. Further study into filtering methods for IMU transient response will need to be performed in order to achieve fully dynamic celestial navigation.

After filtering, a simple first order approximation is applied [31]. A model of the for of Eq. (4.1) was applied. Where A is the amplitude of the approximation, t is time, and τ is a time constant that determines how quickly the system responds.

$$x_o(t) = A(1 - e^{\frac{t}{\tau}}) \tag{4.1}$$

Each trial of the nominal pitch was observed to show the first order behavior. The model chosen to represent that behavior is shown in Eq. 4.2

$$P(X) = P_{nominal} + A(1 - e^{\frac{t}{\tau}}) \tag{4.2}$$

The model of Eq. (4.2) was performed on the IMU pitch measurements of Fig. 4.2. An example of this approximation is shown in Fig. 4.5 for nominal pitch of 10 degrees. The IMU is seen to drift approximately 1.2015 degrees to 11.2015 degrees. A was set equal to 1.2015. Using the 63.2% method and the conversion listed above, the value of τ was found to be 24.557s. This means that the IMU should rest for 4τ or approximately 98.2 seconds uncompensated before attempting to read a new dynamic position. This suggests that data will need to be processed to take into account time lag of the sensor if the camera is to be used in a dynamic application.

Note that there is a strange transient that appears in approximately the first 100 samples. This is likely the product of the moving average filter not having enough data across the initial samples. Further study is needed to see if this small transient signal will affect the celestial navigation camera in a dynamic setting.

The IMU was powered on and collected 10000 pitch measurements while statically oriented at various pitch angles. This test was performed to obtain the first

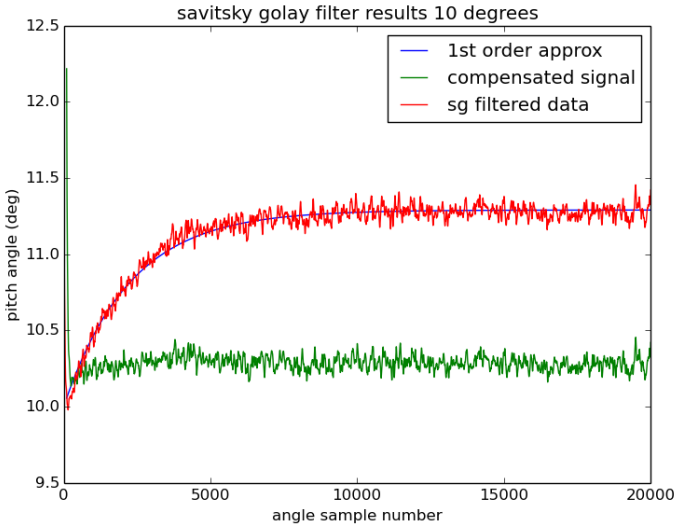

Figure 4.5: First Order Approximation of 10 degree pitch.

order coefficients of Eq. (4.3). In this equation, X represents the pitch measurement from the IMU, $\tau = 2253$ samples, $P(X)$ represents the pitch value as a function of sample X, $P_{nominal}$ is the nominal pitch as measured from the filtered data.

$$P(X) = P_{nominal} + 1.2915(1 - e^{\frac{-X}{2253}}) \tag{4.3}$$

In Fig. 4.5, $P_{nominal} = 10°$. This first order approximation was shown for the trials above to reduce a an approximately 1.3 degree bias to approximately 0.2301 degrees.

4.2 Simulating Experimental Noise and Uncertainty Analysis

It was desirable to determine the uncertainty of the observed estimate H_o of the experimental star tracker. Uncertainty levels for the candidate hardware were used in this calculation [32]. The uncertainty of a function, q, in multiple vari-

ables, $x, x_1, x_2 ... x_n$ is determined by the Pythagorean sum of its partial derivatives multiplied by their respective uncertainties $\Delta x, \Delta x_1, \Delta x_2 ... \Delta x_n$[33].

$$\Delta q = \sqrt{\left(\sum_{i=1}^{n} \frac{\partial q}{\partial x_i} \Delta x_i \right)^2} \tag{4.4}$$

Eq. (4.4) was used in order to determine the uncertainty for the celestial navigation process. A roll correction was derived. This correction rolled the image coordinates $x^{'}, y^{'}$ by an angle θ. This two dimensional rotation matrix is described in Eq. (4.5)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} \tag{4.5}$$

Using Eq. (4.4) normalized to determine the fractional uncertainty, the uncertainty for Eq. (4.5), can be computed to be the following.

$$\Delta y = \Delta x = \sqrt{(\Delta x')^2 + (\Delta y')^2 + (\Delta \theta)^2} \tag{4.6}$$

This leads to the full uncertainty equation of Eq.(4.7).

$$\Delta H_o = \sqrt{2(\Delta x')^2 + 2(\Delta y')^2 + 2(\Delta \theta)^2 + (\Delta \phi)^2} \tag{4.7}$$

Using Eq.(4.7), $\Delta x = \Delta y = 24.2$ arcseconds, $\Delta \theta = \Delta \phi = 0.1°$, the approximate uncertainty was found to be 600 arcseconds. This is only one-sixth of one degree. The scatter plot of position estimates given an assumed location of Durham, NH is presented below in Fig. 4.6

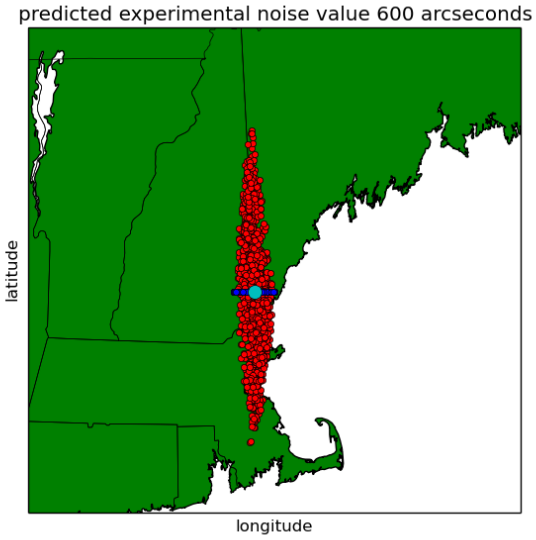

Figure 4.6: Position estimates with 600 arcseconds of uncertainty. Teal represents the true value, Blue is the position estimates with uniform noise, and red is the position estimate with non-uniform noise.

4.3 Experimental Software

The software for the experiment is written in Python 2.7 and can be cloned for personal use from the github.com repository "https://github.com/ckflight/unh-startracker". The Python scripts consist of three major components, each corresponding to the three major components of Fig. 2.10. A block diagram of this software is depicted in Fig. 4.7. The first piece, "Astrometry" is described in detail in the next subsection. Its main purpose is locating stars of interest in the image. It outputs data about the stars, or "sources" in a data dictionary where they can be accessed by another program. In this case, the pixel locations from the data dictionaries are read by CPM.py. CPM.py is a script that performs the center pixel method, determining star locations relative to the image center pixel and assigning an observed altitude. CPM.py also provides other important features such as the

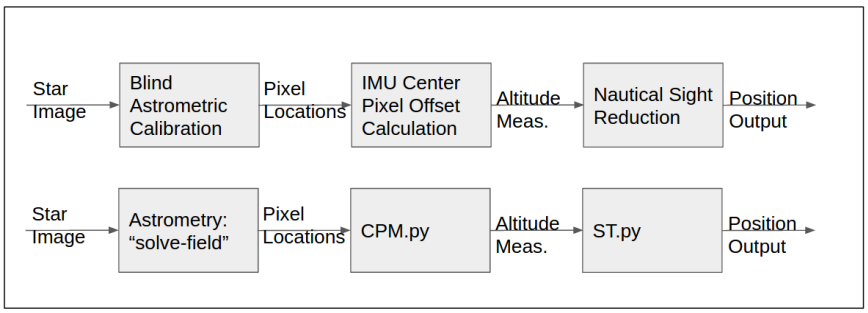

Figure 4.7: Software Block Diagram for Nautical Sight Reduction.

capability to run Astrometry "solve-field" either from the Astrometry.net online server or on a personal computer. Astrometry.net "solve-field" is the command line version of this program capable of identifying sources in the input images Running "solve-field" from a personal computer is much faster. The primary advantage of using the online server is that it displays a result image containing labels of the sources and circles around the sources found in the image. This is very useful for diagnostic purposes. CPM.py also performs basic tasks such as unit conversion and extracting the time a photo was taken from each image file and forwarding it to the next program.

ST.py takes the attitude measurement provided by CPM.py and performs the Nautical Sight Reduction method as described in Chapter 2. ST.py also provides all the math functions used by itself and CPM.py, such as trigonometric functions that take arguments expressed in degrees. Most importantly, ST.py provides a function by which CPM.py can express the y-coordinate of a source star in rectangular or spherical gnomonic projected coordinates. A script known as starmap.py is used to display the maps and histograms of the data. These scripts are available in the Appendices.

4.3.1 Blind Astrometric Calibration

The Astrometry.net software was written by Lang et. al. [25]. The main idea behind the software is to take astronomical photographs and identify astronomical objects, termed "sources" in a generalized lost in space case. Solving the lost in space case means that the stars can be identified simply by their image, and no other input data is necessary. Therefore, the Astrometry.net software uses only data stored in the image pixels, and does not use any metadata such as the size of the image. Source extraction and identification are performed in two parts. The first part, termed "the easy part" by the original authors, is locating the sources in the image. This is performed by an algorithm called SExtractor. This algorithm processes the image to determine a noise level σ and identifies objects who exceed that noise level by 8σ. Further processing is performed to find the centroids of these objects by a Guassian fit. Brightness, from center to edge of the star image typically follows a a Gaussian fit. The results of this process are shown in Fig. 4.8.

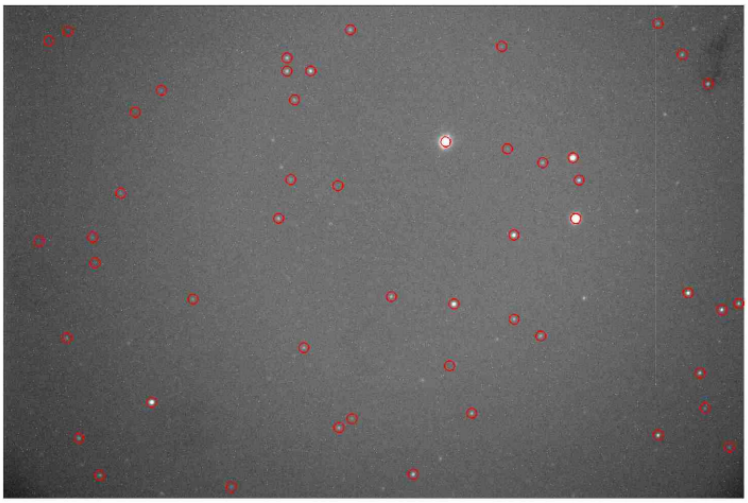

Figure 4.8: Centroid Output of Astrometry.net Software

The next part, termed "the hard part" by the original authors is similar to the method described by Pyramid-LISA, in that "pyramids", or combinations of

four stars, are identified in the image. Pyramids are then compared to a catalog of stars, in this case the catalog used in the USNO-B catalog of stars. This catalog from the U.S. Naval Observatory is suitable for navigation because it catalogs the entire sky to an accuracy of 0.2 arcseconds ($5.5(10^{-5})$ degrees). An example of the pyramids found by the program are shown in Fig. 4.9.

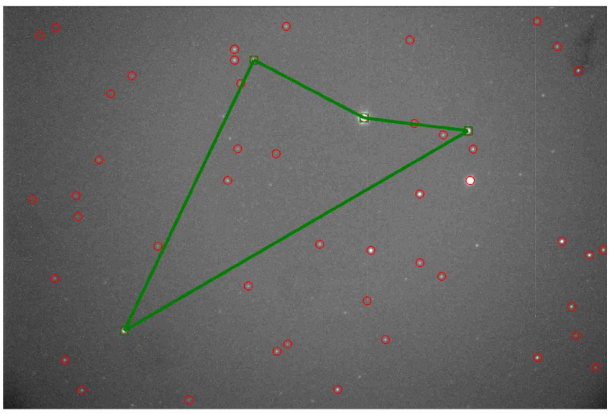

Figure 4.9: "Pyramid" Output of Astrometry.net Software

The key feature of Astrometry, and what makes it the premier software of its kind is that it uses only a single pyramid to determine the starfield in the image. A hash table of the USNO-B catalog is precomputed for use by the algorithm. From there, a hash of the brightest pyramid in the image is compared to the table of hashes to determine where in the celestial sphere the image is taken. The underlying mechanics, such as the actual hash table development, and the Bayesian statistical model by which a determination of which stars are identified by the quad, are beyond the scope of this research and can be found in the literature [29].

4.3.2 Small Angle Approximation

The experimental results provided in this thesis all rely on the small angle approximation such that:

$$sin(\theta) \approx \theta \tag{4.8}$$

$$cos(\theta) \approx 1 - \frac{\theta^2}{2} \tag{4.9}$$

$$tan(\theta) \approx \theta \tag{4.10}$$

Plots of this approximation are shown below in Fig. 4.10 and Fig. 4.11 A plot of

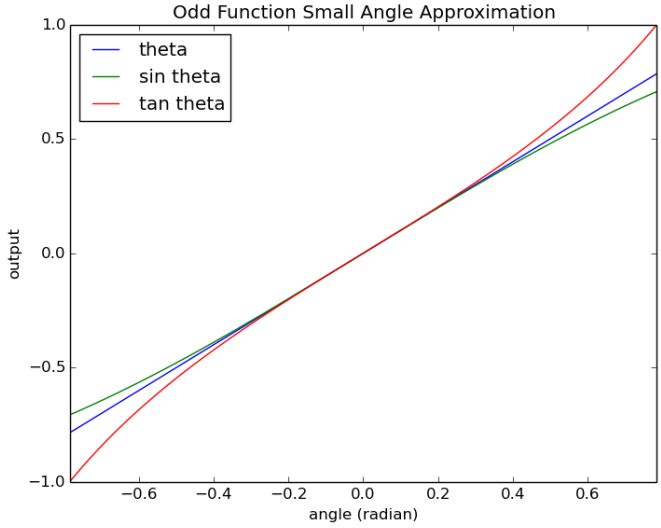

Figure 4.10: Small Angle Approximation of Odd Trigonometric Functions for $\{\theta \in \mathbb{R} | -\frac{\pi}{4} < x < \frac{\pi}{4}\}$

the absolute relative error is shown in Fig. 4.12.

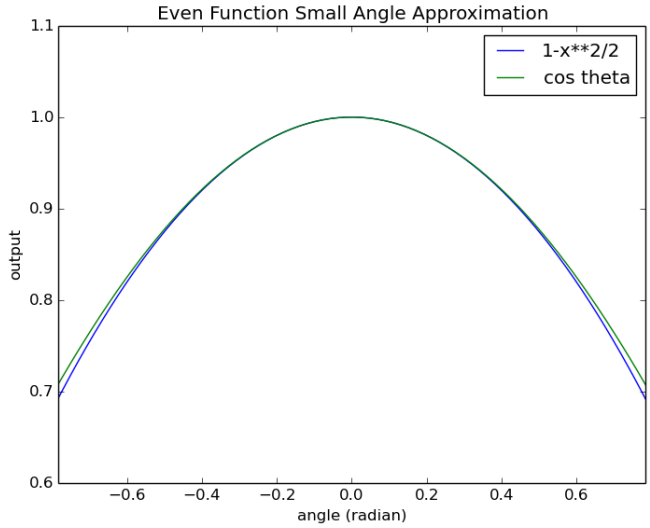

Figure 4.11: Small Angle Approximation of Even Trigonometric Functions for $\{\theta \in \mathbb{R} | -\frac{\pi}{4} < x < \frac{\pi}{4}\}$

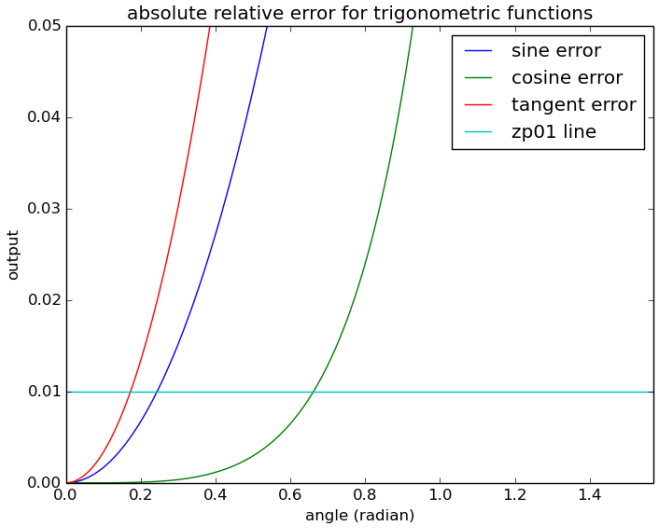

Figure 4.12: Absolute Relative Error with bounding 1% line for Sine, Cosine, and Tangent Functions

A threshold of 1% error was arbitrarily selected in order to ensure that data collected from the star tracker can be input to linear translation and rotation representations for optimization. Since the smallest angle that can be used corresponds to the approximation of the sine function, the maximum angle used in future star tracker datasets was approximately 9.75 degrees. This angular constraint requires that the star observations are closer than 725 pixels to the center pixel. This radius is drawn onto a sample image in Fig. 4.13. Section 4.4.1 will detail the possible

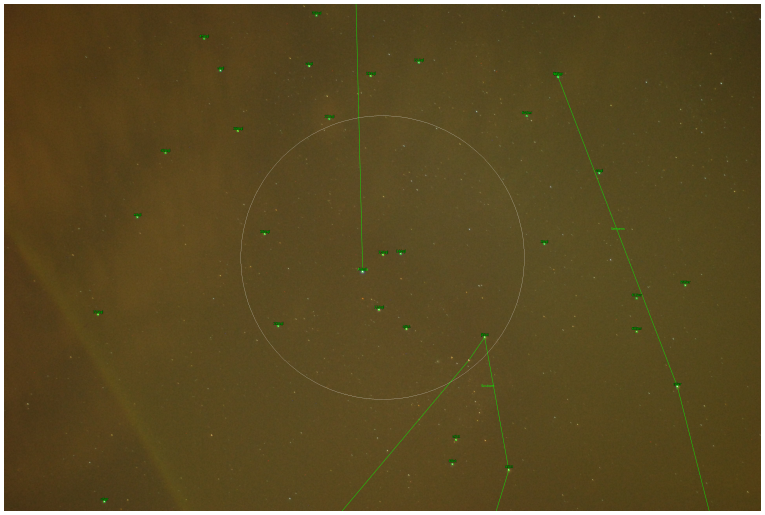

Figure 4.13: 1% Small Angle Approximation Circle for example astronomical photograph

use of the spherical gnomonic projection which seeks to eliminate the need for the small angle approximation.

4.4 Experimental Results

The Nautical Sight Reduction Algorithm was performed on experimental images. Using Astrometry, altitude measurements of star centers were obtained. These measurements were taken from Durham, New Hampshire. With no correction for roll, the mean estimated point is 78.2927W, 44.2405N. The true point is 70.9349W,

43.1337N. This resulted in a mean error of 382.9 nautical miles and can be seen in Figure 4.14. It can be seen from the data that the distribution of points follows a trend similar to those suggested by the uniform noise distribution of Chapter 3.

P_{off}	θ_{off}	Latitude Est.(◦)	Longitude Est.(◦)	Error (◦)
0	0	−78.2927	44.2405	7.4405

Table 4.1: Results for Unbiased Nautical Sight Reduction Algorithm

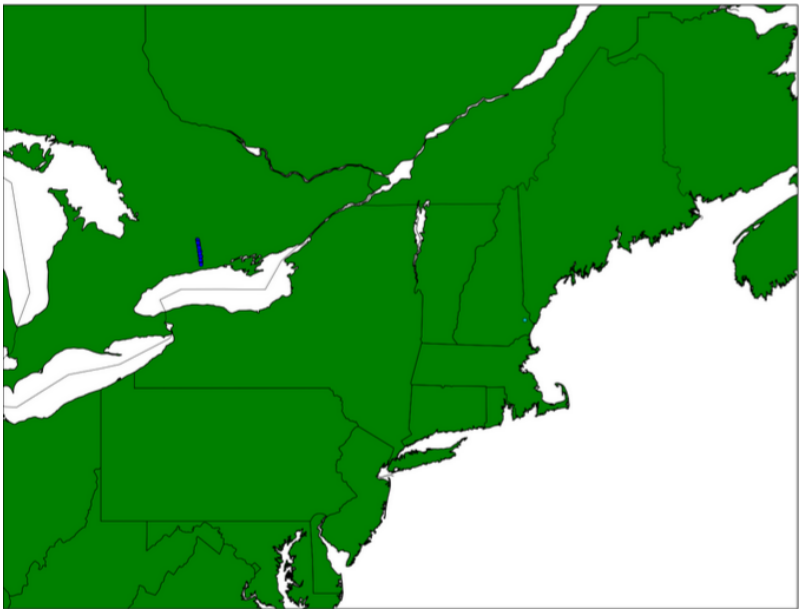

Figure 4.14: Star Camera error with no modifications

A modification of the pitch measurement based on calibration images was added in order to help obtain a more accurate result. This modification is designed to be close to the unmodeled pitch bias. The calibration method attempted to observe a star in the image center pixel and compare H_o and H_c from Eq. (2.62) to determine the bias values. The pitch offset value was -5.109 degrees. The negative sign corresponds to the IMU being pitched too high. The mean estimated point was 71.0236W, 44.2956N. The mean error was 77.09 nautical miles. Error can be

observed in the latitude as time increases. This can be seen in Figure 4.15. These results are summarized in Table 4.2

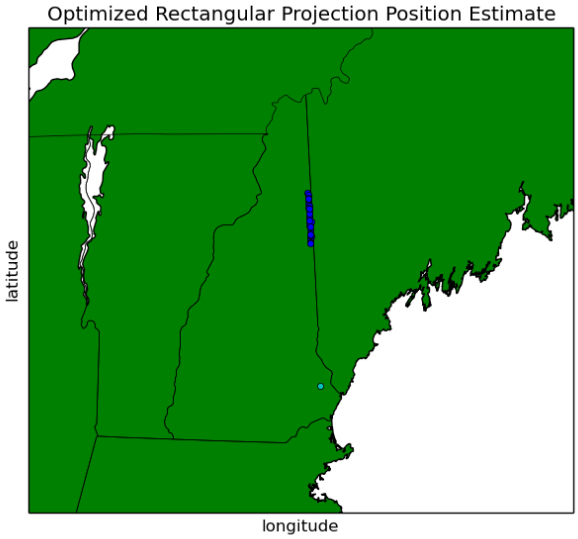

Figure 4.15: Star Camera error with pitch modification

P_{off}	θ_{off}	Latitude Est.(◦)	Longitude Est.(◦)	Error (◦)
-5.109	0	-71.0236	44.2956	1.156

Table 4.2: Simple Pitch Compensation for Rectangular Coordinate Center Pixel Method Cost Function

Arbitrary calibrations of the roll offset led to a more accurate result. An offset of $-2°$ was arbitrarily applied in the roll axis of a sample image. The results are shown in Fig. 4.16, indicating a mean error of approximately 4.5 nautical miles. It was qualitatively apparent that finding a correct roll offset minimizes position estimate error. The method by which roll offset is determined is presented in the following section.

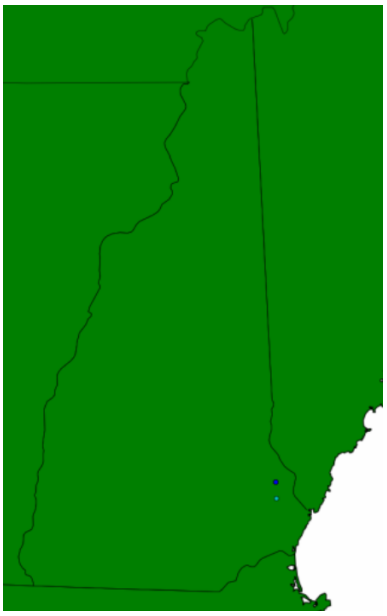

Figure 4.16: Star Camera error with arbitrary roll modification

4.5 Optimization Techniques for Generic Calibration

Pitch and roll offsets cause error in the star camera method. Quantifying these offsets serves to reduce this error. Eq.(4.11) shows the cost function that modeled error as a function of pitch offset, P_{off} and roll offset θ_{off}.

$$J = \sum_{i=1}^{n} \Big(x_i \sin(\theta_{imu} + \theta_{off}) + (y_i + P_{imu} + P_{off}) \cos(\theta_{imu} + \theta_{off}) - (H_c)_i \Big)^2 \quad (4.11)$$

This equation was derived by first translating the initial y-coordinate of the star, y_i by P_{off} for each sample n. Then, the star is rolled in the image by θ_{off}. x_i is the initial x-coordinate of the star. P_{imu} and θ_{imu} are the measured IMU pitch and roll, respectively. Finally, $(H_c)_i$ is the computed altitude of the star, determined by Eq.(2.59). It should be noted that the units of the coefficient terms are in pixels. The units of the argument terms are in degrees.

The cost function of a single observation was plotted in Fig. 4.17 to examine

its qualitative properties. It can be observed that along the roll axis, the behavior

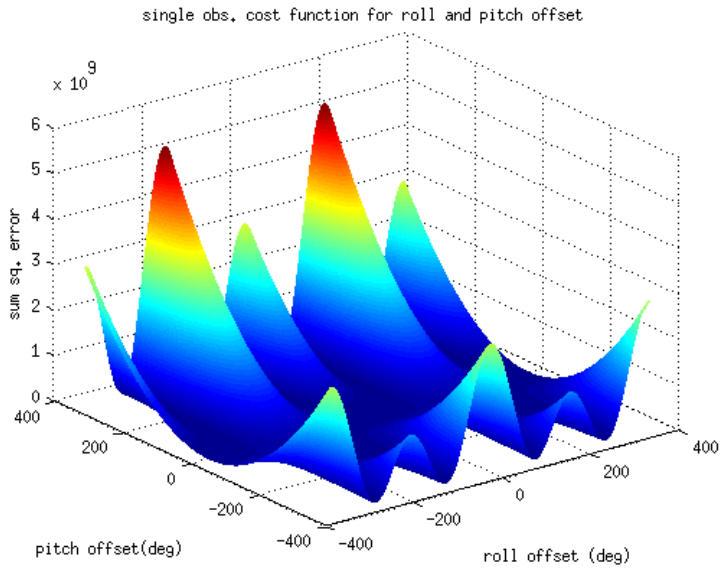

Figure 4.17: Cost Function of a Single Observation

is periodic. This is expected because the roll offset is based on a rotation. Every 360° the trend in the roll repeats itself. On the pitch axis, the error is high for large offsets. Then, as the pitch approaches the true altitude, the error reaches some minimum and increases again as the pitch begins to overshoot. This is an expected result since the plane of the image is currently modeled as Cartesian space. It follows logically that the sum of the cost functions for each observation, depicted in Fig. 4.18, follows the same trends. Note the higher multiplier on the z-axis.

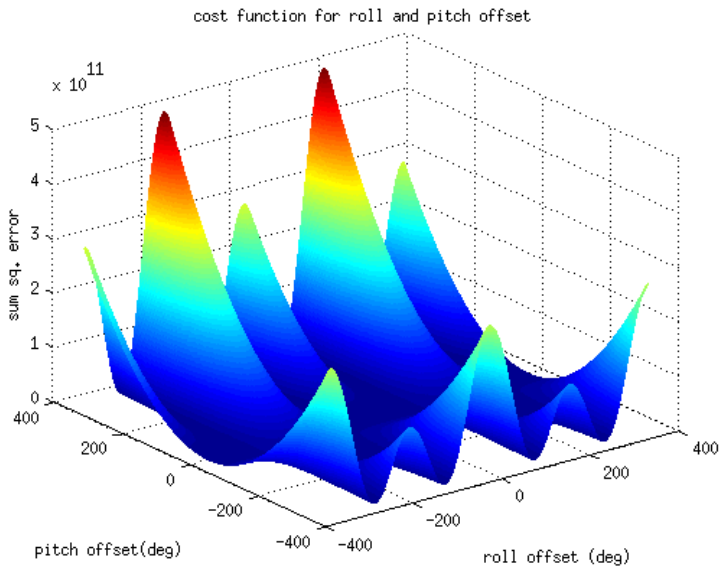

Figure 4.18: Cost Function for entire Lyra Constellation Dataset.

An optimization method was tested to minimize the cost function and characterize the pitch and roll offset. The bounds of the search space are $\pm 15°$. These bounds are fairly lenient and are used to show that lumped error correction is a possibility. Errors above a certain threshold could be from IMU misalignment or poor craftsmanship in the construction. For a production quality star tracker, the optimization would be performed in software after the aforementioned processes were fine tuned. The method used is a parameter space sample and search. This method is simple to implement and determines the minimum value, but is very inefficient because it has to check every element of the resulting matrix.

Parameter Space Search

This algorithm is very simple, at the expense of being very computationally intense. First, an $m \times n$ matrix is computed, where m and n represent the number of elements in each dimension of the parameter space. Then, the function is evaluated at each of those predefined points and placed into an array. Each element of the

array is then iterated over in order to find the minimum value. A script was written in MATLAB to perform this task. The optimization is performed by the following function.

```
[r,c] = find(J==min(min(J)))
```

In this function, J represents the cost function matrix. `Find` and `min` are MATLAB functions that find the indices of a matrix based on certain criteria. In this case they are used to find the minimum value. For a grid of 1000 points that were defined from $\pm 15^{\circ}$ the results found are depicted in Table 4.3. This arrangement of points allows a resolution of 0.03°. The results of this table depict an angular error of approximately 1.268°

P_{off}	θ_{off}	Latitude Est.($\circ$)	Longitude Est.($\circ$)	Error ($\circ$)
-4.9399	-4.7598	-70.634	41.901	1.268

Table 4.3: Optimal Pitch and Roll for Rectangular Coordinate Center Pixel Method Cost Function

The position estimates of sighting of the constellation Lyra on October 9^{th}, 2015 with the optimal roll and pitch added to the calculation result in the map of Fig. 4.19. Notable here is that despite the algorithm being apparently well optimized when examined in terms of degrees, even the 1.268° error found in this method translate into approximately 75 nautical miles of error.

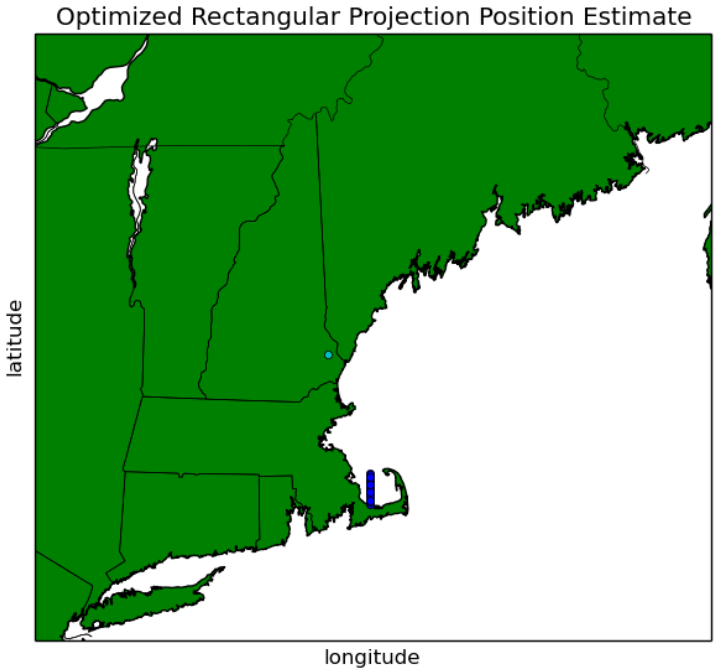

Figure 4.19: Map of Position Estimate Output With Rectangular Center Pixel Method

The heading of this error is noted to be the opposite direction of the test where only the pitch offset was modified. It appears that the roll of the IMU has the greatest effect on the estimate of the latitude. Taking the mean of the estimates calculated by adjusting only the pitch offset and then by adjusting the roll offset gives an improved result depicted in Table 4.4. The result was an error of approximately 5 nautical miles. The position given by this optimization is shown in Fig. 4.20.

P_{off}	θ_{off}	Latitude Est.(∘)	Longitude Est.(∘)	Error (∘)
n/a	n/a	-70.8226	43.147	0.1177

Table 4.4: Mean Position Estimate Tabular Results

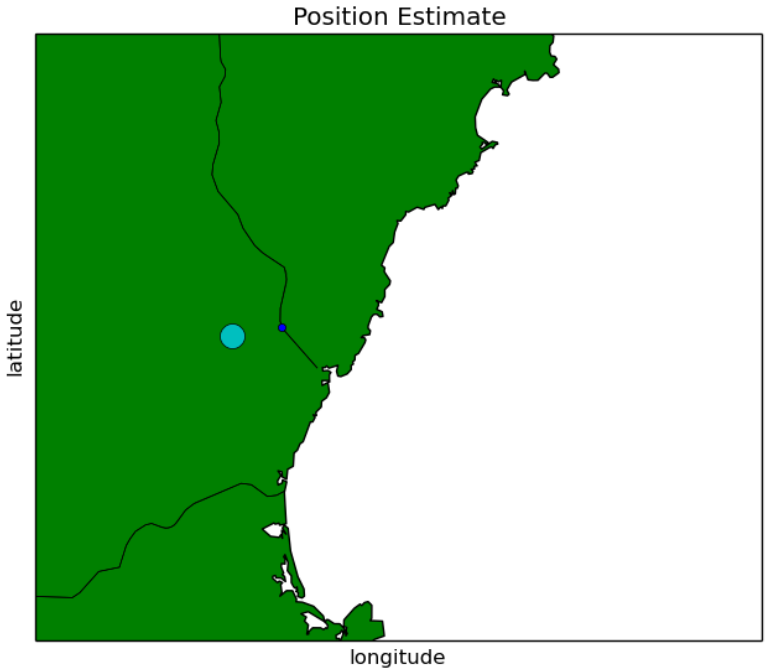

Figure 4.20: Map of Mean Position Estimate Output With Rectangular Center Pixel Method

A slightly improved result is obtained by first optimizing the pitch angle with no roll then optimizing the cost function for pitch and roll, and taking the mean of the two output parameters. The results of this process are shown in Table 4.5. The error of 0.0586° is promising, and is only approximately 3 nautical miles of error. The map of this estimation is shown in Fiq. 4.21.

P_{off}	θ_{off}	Latitude Est.(◦)	Longitude Est.(◦)	Error (◦)
−5.0244	−2.3799	−70.8769	43.1420	0.0586

Table 4.5: Optimal Pitch and Roll for Rectangular Coordinate Center Pixel Method Cost Function

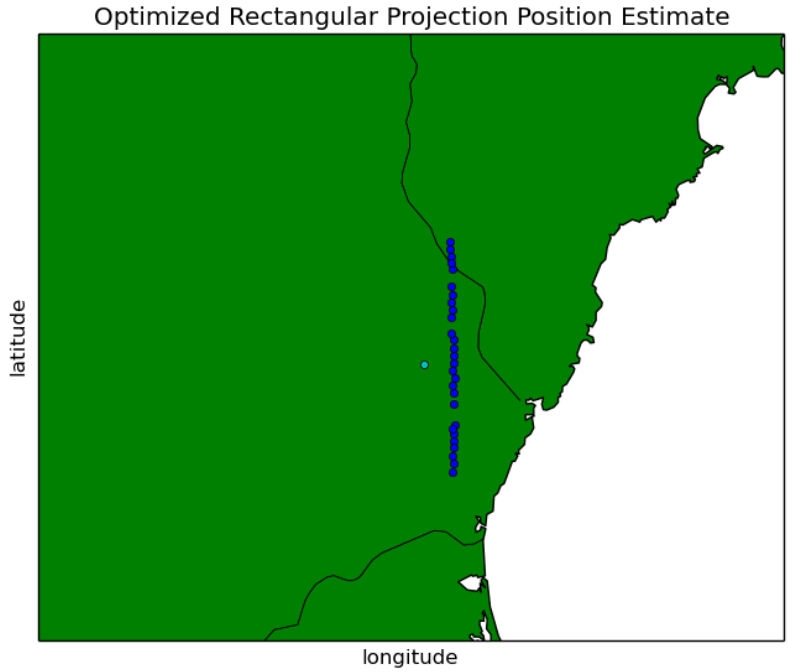

Figure 4.21: Map of Position Estimate Output With Rectangular Center Pixel Method Mean Parameters Method

For further visualization, the mean estimated point is plotted along with the real point on Google Maps, Fig. 4.22. Note that the red pin is the estimated point and the white circle is the true location.

Figure 4.22: Google Map Visualization of Position Estimate Output With Rectangular Center Pixel Method Mean Parameters Method

4.5.1 Optimizing Time Invariant Error

One form of time invariant error is suspected to originate from the requirement that the Celestial Navigation Device adhere to the small angle approximation. The current experimentally implemented version of the center pixel method uses a rectangular approximation. This approximation works best for astrophotography applications with small fields of view. In the case of this celestial navigation device, the field of view is over 30 degrees. The small angle approximation dictates that only a small area of the image be used for legitimate observations. Recall the equations for the small angle approximation from the previous section. One method attempted to refine the time invariant error is the gnomonic projection[34]. This method is expected to aid both the time invariant error and the time varying error because it will eliminate the need to use stars close to the center pixel. Currently, the stars in the observation are selected to be as close to the center pixel as possible in order to adhere to the small angle approximation. The gnomonic projection seeks to eliminate the need for this proximity is described below. The gnomonic coordinate transformation is essential because the light from the star

passes through a spherical lens, onto a CCD element, a flat plane. An example of this distortion can be seen in Fig. 4.23 [34]. The relationship between the spherical

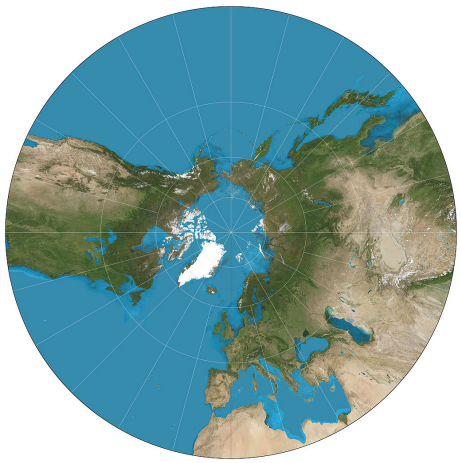

Figure 4.23: Gnomonic Projection Distortion [35]

and planar coordinate frames is given as follows. The center points of the gnomonic projection are ϕ_1 and λ_0 for the horizontal and vertical coordinates, respectively. ϕ represents the horizontal coordinate, and λ represents the vertical coordinate.

$$\phi = sin^{-1}\left(cos(c)sin(\phi_1) + \frac{ysin(c)cos(\phi_1)}{\rho}\right) \tag{4.12}$$

$$\lambda = \lambda_0 + tan^{-1}\left(\frac{xsin(c)}{\rho cos(\phi_1)cos(c) - ysin(\phi_1)sin(c)}\right) \tag{4.13}$$

$$\rho = \sqrt{x^2 + y^2} \tag{4.14}$$

$$c = tan^{-1}(\rho) \tag{4.15}$$

The gnomonic projection modifies the y-coordinate y_i of Eq. (4.11) to the value Y_i of Eq. (4.16). In Eq. (4.16), Y_i is calculated according to the gnomonic projection equations. The value of λ_0 in Eq. (4.13) is defined by the IMU measurement P_{imu}. The horizontal initial coordinate ϕ_1 is set equal to zero. A figure of this new cost function is depicted in Fig. 4.24. The gnomonic projection cost function exhibits the same trends as the rectangular projection. It was observed that the maximum

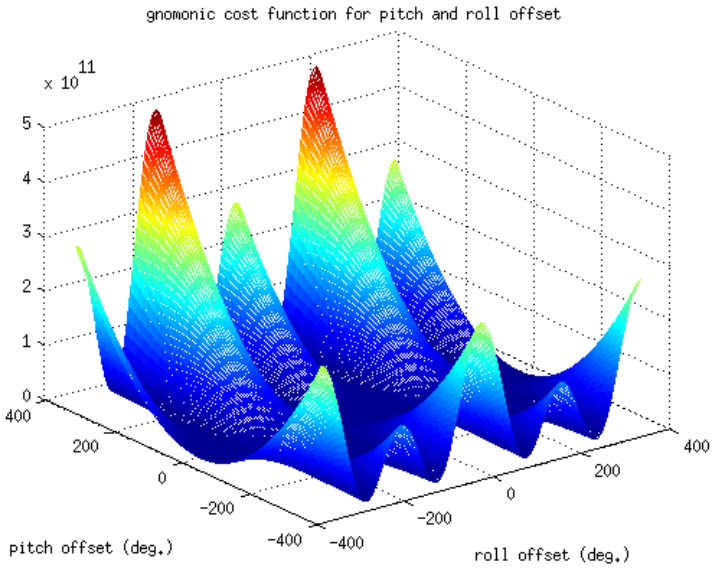

Figure 4.24: Gnomonic Projection Cost Function

value of error was less than the maximum value predicted in the rectangular cost function. The observed values for the sum squared error is $4.924(10^{11})$ for the rectangular cost function compared to $4.908(10^{11})$ for the gnomonic cost function. It was later observed that the optimal point for the gnomonic projection led to higher overall error.

$$J = \sum_{i=1}^{n} \Big(x_i \sin(\theta_{imu} + \theta_{off}) + Y \cos(\theta_{imu} + \theta_{off}) - (H_c)_i \Big)^2 \tag{4.16}$$

The experimental observation with a value of $P_{off} = 0$, and $\theta_{off} = 0$ is depicted in Fig. 4.25. The results are summarized in Table 4.6. It can be seen that the gnomonic projection actually increases the error compared to that of the rectangular approximation. This is unfortunate and likely points to a mistake in the implementation rather than a flaw in the established theory. This method of estimation led to an error of approximately 799 nautical miles. This is about two times the error from the rectangular method, and does not seem to indicate usefulness for navigation.

P_{off}	θ_{off}	Latitude Est.(∘)	Longitude Est.(∘)	Error (∘)
0	0	−83.03213	53.8073	16.1328

Table 4.6: Results of Gnomonic Coordinate Center Pixel Method Cost Function

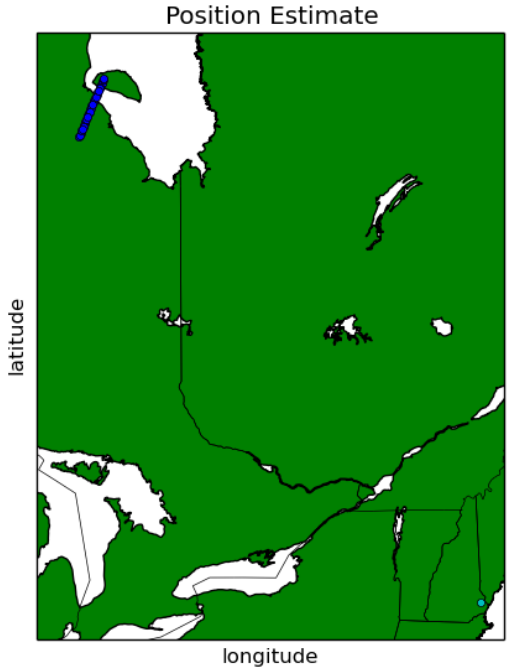

Figure 4.25: Gnomonic Coordinate Center Pixel Method Raw Output.

The optimization of this cost function using the Parameter Search Space results in an optimal $P_{off} = 0.6907°$. The optimal roll offset is $\theta_{off} = -24.9550$. The results of this optimization are shown in Table 4.7. The maps of this optimization are shown in Fig. 4.26. The optimal point disagrees sharply with the same optimization performed on the rectangular center pixel method. Due to the trigonometric and spherical nature of the gnomonic projection, it is likely that the roll angle has a larger effect on the estimates

P_{off}	θ_{off}	Latitude Est.(∘)	Longitude Est.(∘)	Error (∘)
−3.7688	−12.2111	−70.9106	48.1871	5.0534

Table 4.7: Results for "Optimal" Pitch and Roll for Gnomonic Coordinate Center Pixel Method Cost Function

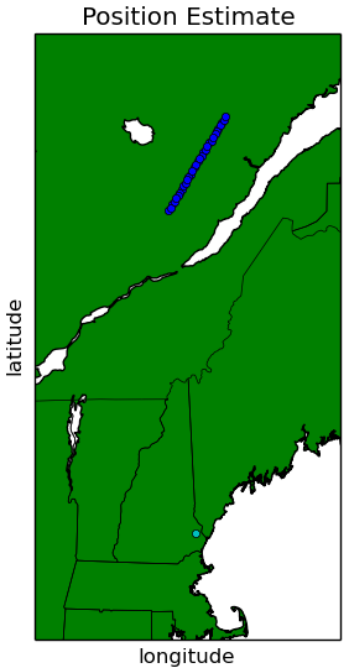

Figure 4.26: Map of Position Estimate Output With Gnomonic Center Pixel Method Mean Parameters Method

4.5.2 Summary of Results

The gnomonic projection appears to be an interesting enrichment problem for star tracking. It was originally expected that the gnomonic projection would reduce estimation error. It can be seen that opposite was achieved by its application. One thing to note is that the star camera does not have an intrinsic requirement to be able to see the entire star field. These results suggest that a viable modification

to the design might be to use a telescoping lens that mechanically guarantees adherence to the small angle approximation.

Cost function analysis indicates that the best way to minimize the star camera error is to use the rectangular center pixel method to optimize over the pitch, followed by optimizing over the pitch and roll and taking the mean of the output parameters. It is suspected that this method would fit nicely into a Mean Squared Error (MSE) cost function representation [36].

Chapter 5

Discussion and Future Work

In this section, the discussion is broken into four major components.
1. The star camera in this research is compared to other established methods.
2. The zenith requirement of the Attitude Matrix Formulated Methods is explained in the context of the presented star camera.
3. The effects of error on the surfaces of other planets are explained.
4. Analysis is performed on the time varying/noise properties of the measurement of H_o and H_c.
Future work is discussed, including suggestions to improve both static accuracy and make dynamic navigation possible.

5.1 Comparison to Established Methods

This research was designed in order to give the UNH Advanced Controls Laboratory the capability to statically determine latitude and longitude on Earth based on an image of the stars. The evidence presented in this thesis suggests that accurate position determination is possible via the use of a commercial digital camera and commercially available IMU. The minimum achieved error using the rectangular cost function is 0.0586°. This corresponds to an error of approximately 3 nautical

miles. In the introduction, CST, the Modified Attitude Matrix Formulation, the Phobos Transit Method, and the DSN are presented. Fig. 5.1 shows the unbiased star camera error as compared to the other established methods. Fig. 5.2 shows the rectangular mean parameters optimized star camera as compared to the other established methods. It can be seen from the plots that for the unbiased, un-

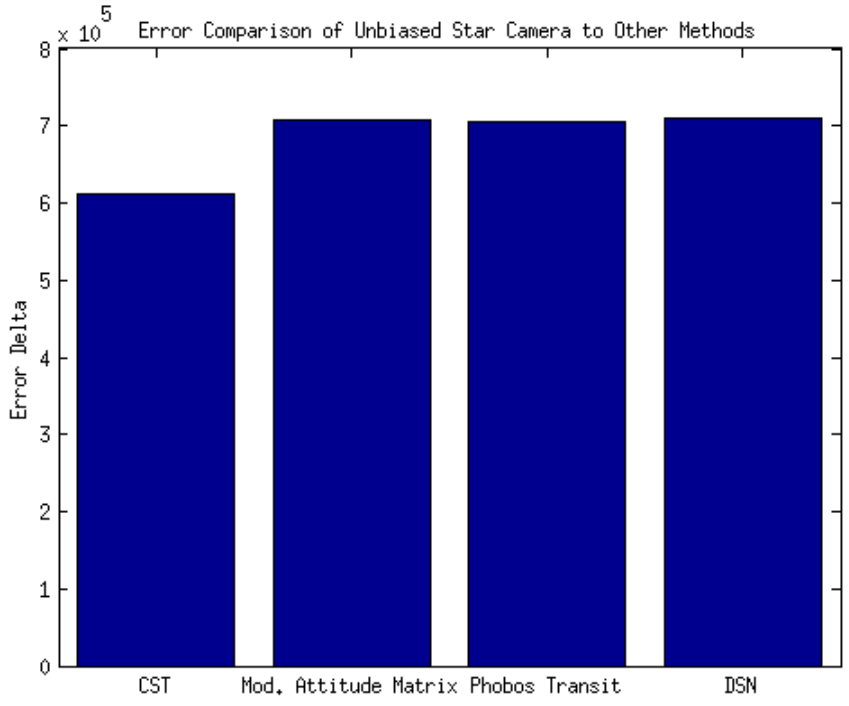

Figure 5.1: Error of the Unbiased Star Camera Compared to Established Methods

modified star camera, the error is much greater than that of the other established methods. With optimization, this changes. The presented star camera has an error circle approximately 90km smaller than CST. Although the error of the optimized star camera still appears to be greater than those of the Modified Attitude Matrix Method and the Phobos Transit method, it should be noted that those are theoretical errors only. In the case of the Modified Attitude Matrix Method, the

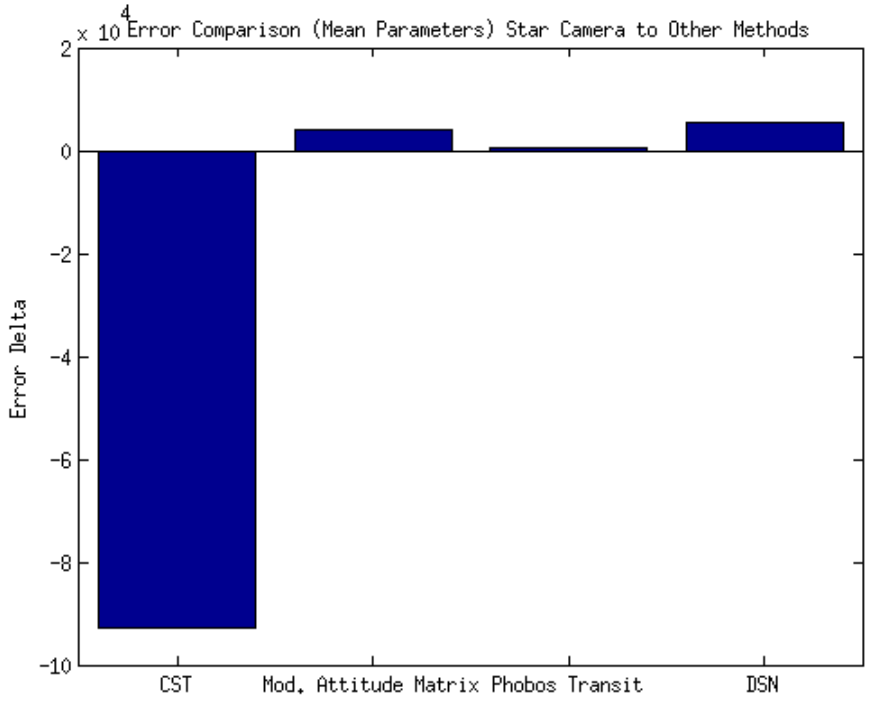

Figure 5.2: Error of the Rectangular Mean Parameters Star Camera Compared to Established Methods

error bar was developed with the use of the backtracker method. The backtracker method is an interesting theoretical development but needs to be improved to be for practical use on an extraterrestrial surface. The Phobos Transit method has a similar caveat in that the error was simulated with various levels of noise. The Phobos Transit Method a fundamental idea similar to the star camera presented in this work, so it is expected that they both would have similar magnitudes of error. The obvious choice for accurate position determination on an extraterrestrial surface mission is still DSN. In this research, by using improved sensors and optimization, surpassing DSN is within the realm of possibility.

One other feature of the star camera is that it is self contained and thus does not require much supporting infrastructure. Other than DSN, the presented

method and the other established methods of celestial navigation are self contained. However, DSN has by far the best error circle. The error circle plotted by a selection of celestial navigation methods is shown below in Fig. 5.3. In this figure, legend obscures part of the plot but the data is still circular. This first plot is a good demonstration about why lowering the error is important. The possible position may be known to a certain error radius but that means that, in general, the actual position could be anywhere within that circle. It can be seen that the error circle for the unbiased star camera dwarfs the other error circles.

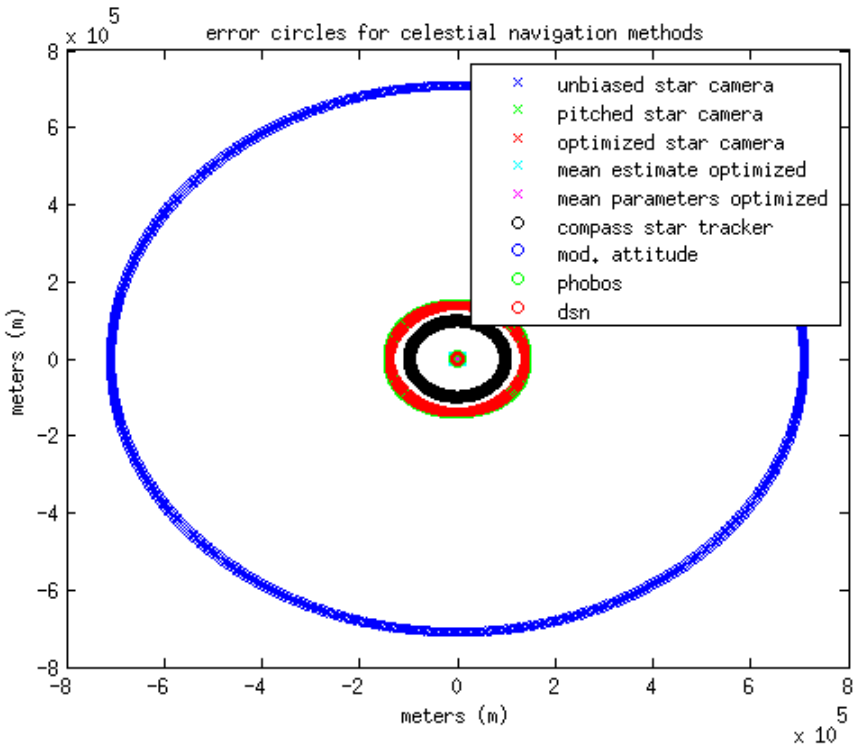

Figure 5.3: Error Circles for Star Camera and Established Methods.

A closer view is shown in Fig 5.4 to demonstrate the scaling effect of the smaller error circle. Although the legend obscures part of the plot, the data is still circular. Even in the zoomed-in version of this plot, DSN is still by far the best at determining position, and the presented star camera performs about as well as the

theoretical implementations of the Modified Attitude Matrix and Phobos Transit Methods. To further demonstrate the value of a small error circle, DSN estimates the location of an object to an area 0.14% the size of the area estimated by the star camera mean parameters method.

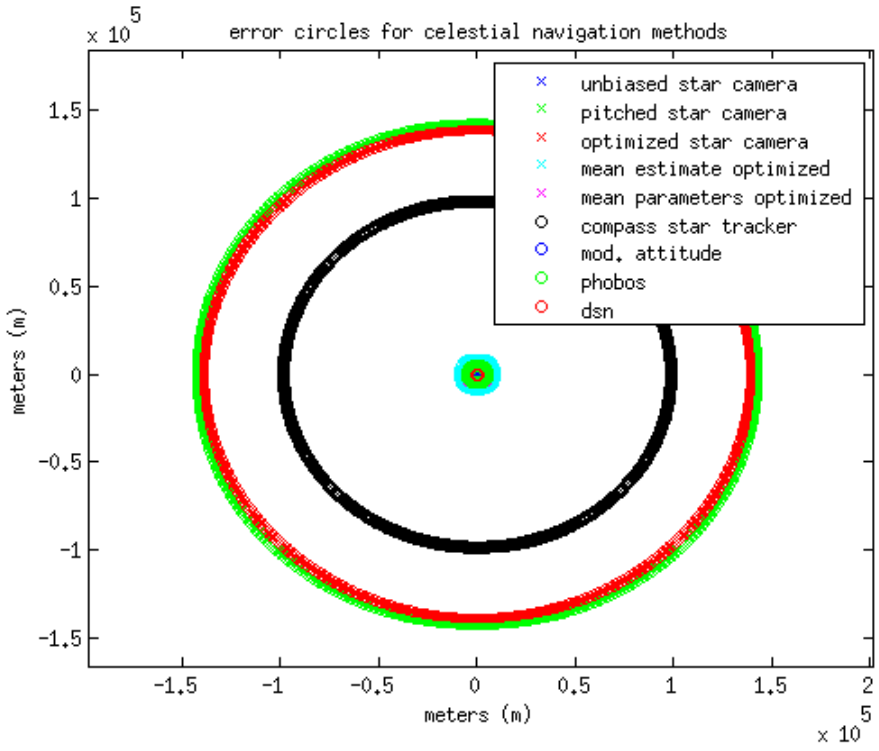

Figure 5.4: Closer Error Circles for Star Camera and Established Methods.

5.2 Zenith Requirement

One concern of the attitude matrix formulated methods is that they had a strict zenith orientation requirement. This requirement says that the camera must be oriented vertically upwards to comply with the equations defining those algorithms. This work seeks to reduce that requirement. Theoretically in the Nautical Sight Reduction Algorithm, any pitch pointing direction up to 80° can be used to de-

termine a position estimate. This restriction is imposed by the equations of the Nautical Sight Reduction Method [16]. The measured and computed altitude output from the star camera is shown in the following tables for each of the three observational formats of Chapter 3. The tables compare the observed altitudes, H_o with the computed altitudes, H_c of the selected stars in the unbiased simulations with no noise. Recall again that these values are defined in Eq. (2.62). Table 5.1 corresponds to the Case 1 observational format. Table 5.2 corresponds to the Case 2 observational format. Table 5.3 corresponds to the Case 3 observational format. It can be seen that altitude measurements from 25.4899° to 59.2972° were all observed to give correct position estimates under perfect conditions. This indicates that there is greater freedom in the direction the camera can be pointed for accurate navigation compared to the other established methods.

.	Regulus	Antares	Kochab
H_o	27.3054	25.4899	47.8823
H_c	27.3054	25.4899	47.8823

Table 5.1: H_o and H_c for Case 1 Observational Format.

.	Sulafat 1	Sheliak 2	Vega
H_o	59.2972	58.0256	57.9522
H_c	59.2972	58.0256	57.9522

Table 5.2: H_o and H_c for Case 2 Observational Format.

.	Altair 1	Altair 2	Altair 3
H_o	54.1453	53.5746	52.9904
H_c	54.1453	53.5746	52.9904

Table 5.3: H_o and H_c for Case 3 Observational Format.

5.3 Time Variation of $H_o - H_c$

It was expected that error would increase with increased length of observation. The reason for this is that the current calculation of GHA and LHA uses linear interpolation. Better ephemeris data may help correct this error. Fig. 5.5 shows the plot of $p = H_o - H_c$, a filtered version using a weighted average filter, and linear regression plots of each one. This plot confirms that error increased over time. The mean error of the unfiltered signal is approximately $-0.4933°$, and the standard deviation is approximately 0.9941. The mean error of the filtered signal is -0.4231 and the standard deviation is 0.3007. The lines of best fit fit the form $y = mx+b$ with $m = -0.0113$ and $b = -0.2332$ for the unfiltered linearization, and $m = -0.0116$ and $b = -0.2985$ for the filtered linearization. These results indicates that a filter on the pixel locations of the image may increase the accuracy of the star camera. They also indicate that from the beginning of the dataset, $H_o - H_c$ is less than expected. There are several possible reasons behind the inaccuracy of the star camera. This is an interesting topic for future research.

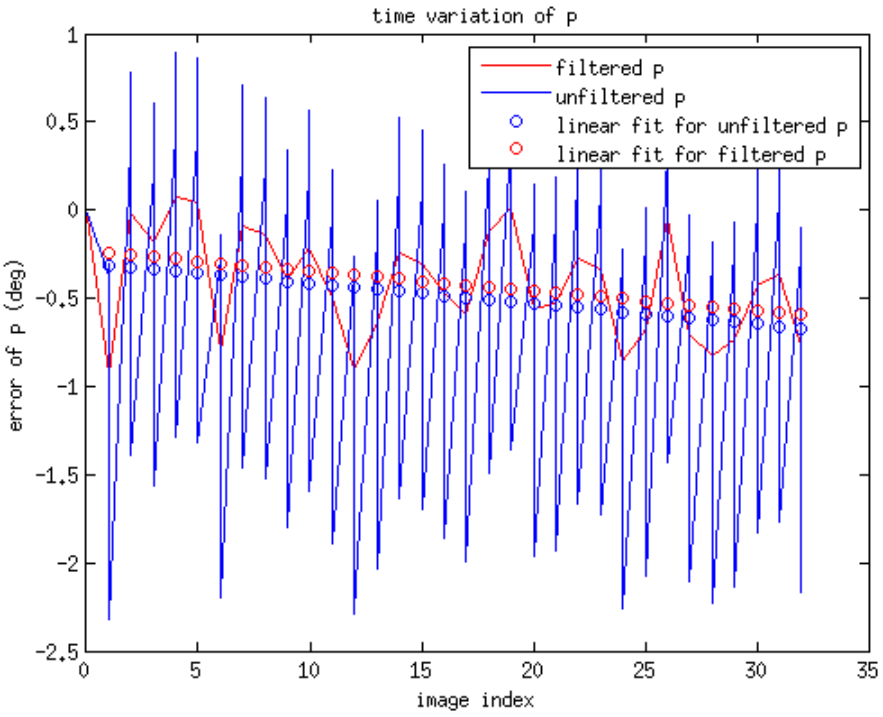

Figure 5.5: Linear regression of filtered and unfiltered $H_o - H_c$

5.4 Use on Extraterrestrial Surfaces

Position Estimate Error

It is notable that the position estimate of the presented star camera is expressed in degrees and then converted into a linear distance such as meters or nautical miles. This is an important consideration because the linear distance is a function of the radius of the planet. The distance d between two points (δ_1, λ_1), (δ_2, λ_2) on a sphere or globe of radius a is a great circle [37]. This is shown in Eq. (5.1).

$$d = a\cos(\cos(\delta_1)\cos(\delta_2)\cos(\lambda_1 - \lambda_2) + \sin(\delta_1)\sin(\delta_2) \tag{5.1}$$

For a pure translation in latitude $(\delta_1, \lambda_1) = (0, error)$ from the point $(\delta_2, \lambda_2) = (0, 0)$ across the planet equator, the equation is simplified to the circular arc length

such that:

$$d = a\lambda_1 \tag{5.2}$$

Fig. 5.6 shows how the error changes as a function of radius for different optimization schemes of the presented star camera. In general, it is observed that if the radius of the planet is smaller, then the linear distance will also be smaller for a given error offset.

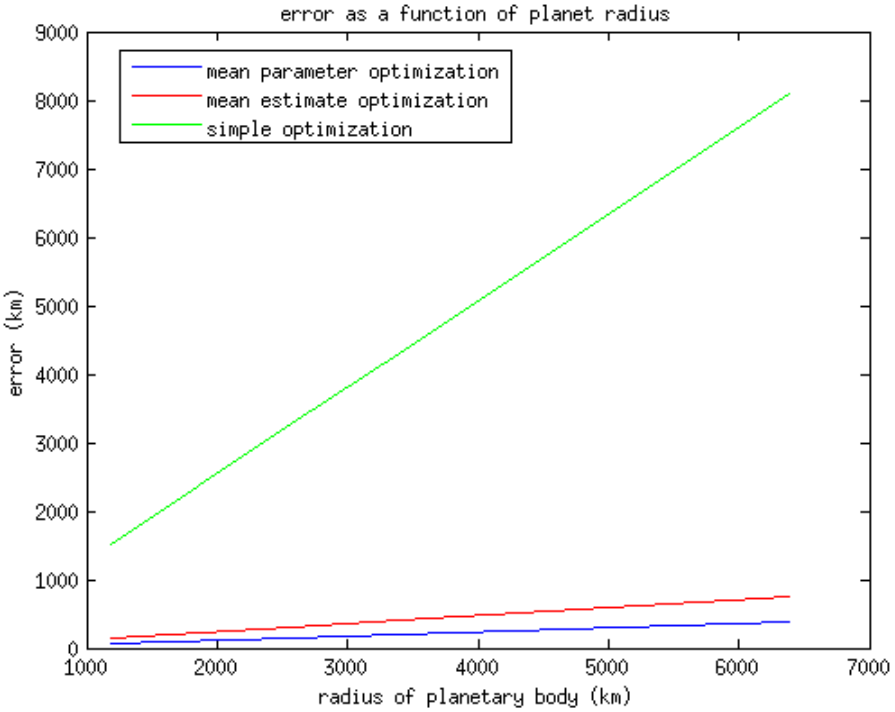

Figure 5.6: Error of the Unbiased Star Camera Compared to Established Methods

5.5 Future Work

Several important contributions were achieved and discussed in this thesis. Many more remain in order to transition from this proof-of-concept device to a viable

solution for dynamic celestial navigation. In general, these improvements to the design fall into two categories. The first is those that increase the star tracker static accuracy. The second are those that make dynamic navigation a possibility.

Static Accuracy

The static accuracy of the camera should be perfected before moving on to dynamic navigation. More translational tests would ensure robustness. It remains to be seen if the methods used to optimize the Lyra observation data set will hold across many other observations. Another test that would be interesting to perform would be a static test that views different parts of the sky from the same point and ensured that no matter the heading direction was observed, the Nautical Sight Reduction Algorithm would yield the correct position. Further analysis should be performed to reduce the amount of noise in both the IMU, Fig. 4.5, and the pixel location of the star on the image, Fig 5.5.

Dynamic Accuracy

Although the presented star camera seems to be performing celestial navigation to varying levels of success, depending on the optimization method applied, much more development is neccesary before dynamic navigation can be realized. The first challenge is reducing of time that estimating the position will take. Each image has an 8 second exposure time, followed by a processing time of up to 15 seconds for determining the estimate. This means that there could be up to 23 seconds between estimates of the location. Second, a platform by which the star camera can stay fixed on the sky, in the presence of any disturbances that may be encountered during an extraterrestrial mission, will also be necessary. Dynamic use of the IMU should be investigated as currently, the IMU data is read from a file to perform post processed position estimation.

Several changes should be made to the overall design in order to achieve full

autonomy. Firstly, the images are stored on an SD card, where they are then transferred to a computer for post processing. This makes dynamic navigation impossible under the current setup since the determination of position occurs *a posteriori*. This could be accomplished by the use of a single board computer such as a Raspberry Pi[38] or BeagleBone Black[39].

Chapter 6

Conclusions

There were three main objectives of this research:
1. Develop a proof-of-concept experimental device that can use an image of the stars and output meaningful position estimate.
2. Demonstrate the ability of this device to use a wide range of zenith angles.
3. Provide analysis on the performance of the individual sensors and overall accuracy of the device.

A device was developed that uses the Nautical Sight Reduction Algorithm. The development of this device was motivated by the desire to have a viable celestial navigation experiment. That image is fed into the Blind Astrometric Calibration software package which outputs the sources - "stars" in the image. The pixel locations of those stars are input to a software which assigns IMU values to the center pixel of the image, and calculates the altitude. The altitude is then used as an input to the Nautical Sight Reduction Algorithm. Plots of the position estimates are shown throughout the document. The raw position estimate was approximately 382 nautical miles of error. This error was reduced by a series of optimization methods, namely cost function analysis to approximately 3 nautical miles. It was found that the rectangular coordinate method of implementing the center pixel method gave a much more accurate position estimate compared to the

gnomonic method. This was an unexpected result. Possible continuations include refining the implementation of the gnomonic method, or restricting the field of view to mechanically ensure adherence to the small angle approximation.

The ability of this device to use a wide range of zenith angles was developed primarily by examination of simulated analytic results. It was shown that for three different observational formats, a large range of altitudes could be used. The star camera and associated simulations observed altitudes ranging from 25.4899° to 59.2972°. Experimental observations of the constellation Lyra supported this trend, with measured altitudes in the range of approximately 57° to 59°.

Analysis on the IMU found that the measurements obtained were biased and followed the trend of a first order system. A simple first order model was developed as a compensation method for this bias and showed that if applied in a dynamic system, could possibly reduce this error from approximately 1.20° to 0.2301°. Analysis of the small angle approximation produced a bounding line by which images are recommended to follow. This boundary suggests viewing stars which are no greater than 725 pixels away from the center pixel. If further stars are used, then large errors may result. The method of celestial navigation presented in this research was compared to other established methods in the field of celestial navigation. It was found that compared to the Compass Star Tracker, it is possible for a much smaller error to be realized. This figure also shows that compared to the state-of-the-art navigation method, the Deep Space Network, the presented device has a far greater error. Most importantly perhaps, an experimental platform for position determination has been realized for use in future applications.

Bibliography

[1] "Luna 2." *Wikipedia.* Wikimedia Foundation, 28 Apr. 2016. Web. 03 May 2016.

[2] *Surveyor: Program Results.* Washington, D.C.: National Aeronautics and Space Administration, Scientific and Technical Information Division, 1969. Print.

[3] Administration, National Aeronautics And Space. "Mars Polar Lander/Deep Space 2." *Mars Polar Lander/ Deep Space 2* (n.d.): n. pag. Jet Propulsion Laboratory. NASA. Web. 04 May 2016.

[4] "MESSENGER FAQ." *Applied Physics Laboratory.* Johns Hopkins, 2015. Web. 04 May 2016.

[5] Mitchell, Don P. "Soviet Telemetry Systems." *Russian Telemetry Systems.* Mental Landscape, 2003. Web. 03 May 2016.

[6] "Deep Space Network." (n.d.): n. pag. *Deep Space Network.* Jet Propulsion Laboratory. Web.

[7] "Navigation - Mars Science Laboratory." Navigation - Mars Science Laboratory. NASA, n.d. Web. 03 Feb. 2016.

[8] United States. NASA. JPL. *210 Delta Differential One Way Ranging.* By James S. Border. N.p.: n.p., n.d. California Institute of Technology, 4 Dec. 2014. Web. 3 Feb. 2016.

[9] Trautner et. al. "A New Celestial Navigation Method For Mars Landers". *Lunar and Planetary Science XXXV (2004).*

[10] Malay, Benjamin P. *Celestial Navigation on the Surface of Mars.* No. USNA-1531-2. NAVAL ACADEMY ANNAPOLIS MD, 2001.

[11] Samaan, Malak A., Daniele Mortari, and John L. Junkins. "Compass star tracker for GPS applications." *Advances in the Astronautical Sciences* 118 (2004): 75-86.

[12] Swanzy, Michael John. *Analysis and Demonstration: a Proof-of-Concept Compass Star Tracker.* Diss. Texas A&M University, 2007.

[13] Thein, M. L., David A. Quinn, and David C. Folta. "Celestial navigation (CelNav): lunar surface navigation." *Proceedings of the 2008 AIAA/AAS Astrodynamics Specialist Congress and Exposition.* 2008.

[14] Perkins, Jared. *Development of the NASA Celestial Navigation Method For Dynamic Extraterrestrial Surface Navigation.* Diss. University of New Hampshire, 2014.

[15] Umland, Henning. "A Short Guide to Celestial Navigation." *A Short Guide to Celestial Navigation.* N.p., 28 Feb. 2016. Web. 10 Mar. 2016.

[16] *Nautical Almanac 2016.* N.p.: Paradise Cay Pubns, 2015. Print.

[17] Fuller, Thomas C.K. "Celestial Navigation Device for Future Autonomous Applications." 2016 Spaceflight Mechanics Conference. California, Napa. 18 Feb. 2016. Lecture.

[18] Johnson, Michael Andrew, and May Win L. Thein. "The Analytical Study of Particle Swarm Optimization Approaches for Extraterrestrial Surface Navigational Searches." *AIAA/AAS Astrodynamics Specialist Conference.* 2014.

[19] Underwood, Amy R., and M. L. Thein. "A Comparative Study of Celestial Navigation-Based Controller Designs for Extraterrestrial Surface Navigation." *Proceedings of the AIAA Guidance, Navigation, and Control Conference (GNC'13).* 2013.

[20] Pedley, Mark. "Tilt sensing using a three-axis accelerometer." *Freescale Semiconductor Application Note* (2013): 2012-2013.

[21] Mortari, Daniele, et al. "The pyramid star identification technique." *Navigation* 51.3 (2004): 171-183.

[22] O'Neil, Peter V. *Advanced Engineering Mathematics.* Stamford: Cengage Learning, 2012. Print.

[23] Knight, Randall D. *Physics with Modern Physics; Scientists and Engineers; a Strategic Approach.* Pearson Education Inc.: n.p., 2013. Print.

[24] Vallado, David A., and Wayne D. McClain. *Fundamentals of Astrodynamics and Applications.* Dordrecht: Kluwer Academic, 2001. Print.

[25] Lang, Dustin, et al. "Astrometry. net: Blind Astrometric Calibration of Arbitrary Astronomical Images." *The Astronomical Journal* 139.5 (2010): 1782.

[26] Van Leeuwen, Floor. "Validation of the new Hipparcos reduction." *Astronomy & Astrophysics* 474.2 (2007): 653-664.

[27] Blewitt, Mary, and Thomas C. Bergel. *Celestial Navigation for Yachtsmen.* Camden, Me.: International Marine, 1995. Print.

[28] Inf 5440 - Cmos Image Sensors. *CHARACTERIZATION of CMOS Sensor* (n.d.): n. pag. University of Oslo. UO. Web. 18 Feb. 2016.

[29] Lang, Dustin. *Blind Astrometric Calibration of Arbitrary Astronomical Images* (n.d.): n. pag. PPenteado. Paulo Penteado. Web. 15 May 2016.

[30] Schafer, Ronald W. "What is a Savitzky-Golay filter?[lecture notes]." *Signal Processing Magazine, IEEE 28.4* (2011): 111-117.

[31] Ogata, Katsuhiko. *System Dynamics.* Upper Saddle River, NJ: Pearson/Prentice Hall, 2004. Print.

[32] Devices, Inc. Analog. ADXL345 (Rev. 0) (n.d.): n. pag. *Razor 9-DOF IMU.* Razor. Web. 19 May 2016.

[33] "Error Propagation." Appalachian State University Dept. of Physics. Appalachian State University, 2015. Web. 06 May 2016.

[34] "Gnomonic Projection." – from *Wolfram MathWorld.* N.p., n.d. Web. 03 Feb. 2016.

[35] "Gnomonic Projection." *Wikipedia.* Wikimedia Foundation, 11 Mar. 2016. Web. 19 May 2016.

[36] Raschka, Sebastian. *Python Machine Learning.* Birmingham: Packt Limited, 2015. Print.

[37] Weisstein, Eric W. "Great Circle." From MathWorld–A Wolfram Web Resource.

[38] "Raspberry Pi." *Raspberry Pi.* Raspberry Pi Foundation, n.d. Web. 17 May 2016.

[39] "BeagleBone." *Beagleboard.org.* N.p., n.d. Web. 17 May 2016.

Appendices

Appendix A

Nautical Sight Reduction Script

The code implements the Nautical Sight Reduction Algorithm in Python 2.7. It also contains math libraries used throughout the code. If it is desired to use these functions, other scripts can import them.

```
# UNH Star Tracker Calculator
# Uses the Nautical Almanac "Direct Computational" Method
# 2015 Nautical Almanac Commercial Edition pg. 283
# tckf - september - 2015
from __future__ import division
import math as m
import random as rng
import numpy as np
#simple math utils for working in degrees
def sind(theta):
    return m.sin(m.radians(theta))
def cosd(theta):
    return m.cos(m.radians(theta))
```

```
def tand(theta):
    return m.tan(m.radians(theta))

def asind(arg):
    return m.degrees(m.asin(arg))
def acosd(arg):
    return m.degrees(m.acos(arg))
def atand(arg):
    return m.degrees(m.acos(arg))

# slightly harder math for determining the gnomonic projection
def gnomonicProjection(x,y,lam0):
    # takes in xy coordinates and a starting lambda and
    # outputs the gnomonic projection coordinate
    # equivalents.
    rho = np.sqrt(x**2 + y**2)
    c = np.arctan2(rho,1)
    lam = lam0 + np.arctan2(x*np.sin(c), rho*np.cos(c))
    return lam
# doing a bunch of different logs
dLog = open('dLog', 'w')
dLogFinalPoint = open('dLogFinalPoint', 'w')
# for the parameters that aren't supposed to change on each pass
inertialParams = open('inertials0', 'w')

# for the parameters that are changing each iteration
changingParams = open('changes0', 'w')

ip1 = open('inertials1', 'w')
```

```
cp1 = open('changes1', 'w')

ip2 = open('inertials2', 'w')
cp2 = open('changes2', 'w')

analysis = open('analysis.csv', 'w')
inertials = open('trueInertials.csv', 'w')
# to test this we're goign to use "regulus" to calculate the location

class Sextant:

    # the sextant class deals with computing the Observed altitude from the
    # sextant measurement

    def __init__(self,hSextant, eyeHeight):
        # this is the value you measure from the starCamera or sextant
        self.hSextant = hSextant        # degrees
        self.eyeHeight = eyeHeight      # meteres
        self.dip = self.calcDip()
        self.hApparent = self.calcApparentAlt()
        self.r = self.calcRefraction()
        self.Ho = self.calcAltitude()

    def calcDip(self):
        # calculates an error based on your height of eye
        D = 0.293*m.sqrt(self.eyeHeight)     # degrees
        return D

    def calcApparentAlt(self):
```

```
        Hs = self.hSextant - self.dip
        return Hs

    def calcRefraction(self):
        # refraction is based on
        H = self.hApparent
        Ro = (0.0167)/(tand((H + 7.32)/(H + 4.32)))
        return Ro

    def calcAltitude(self):
        H = self.hApparent
        Ro = self.r
        Ho = H - Ro
        return Ho

    def printAns(self):
        print self.hSextant        # degrees
        print self.eyeHeight       # meteres
        print self.dip
        print self.hApparent
        print self.r
        print self.Ho

class Observation:
    # LF is LONGITUDE
    # BF is LATITUDE
    # this class defines the observations that we get from the sextant or CNC
    # CNC = CelNavCamera.  PRETTY SWAG
```

```
# shared among the rest of the instances of this object
fixDate = [2015, 10, 7]      # (year, month, date) not used
#fixTime = [21, 0, 0]        # (hour, minute second)
v = 0                       # knots dont edit bc we're statis
track = 0                  # track heading in degrees
lf = -70.934916                  # est. long, @ fixTime E+ W-, degrees
bf = 43.13376                   # est. lat @ fixTime N+ S-, degrees
#lf = -15         # this one is for the example problem
#bf = 32
#lf = -75
#bf = 45 # note the sign convention for lha is opposite

def __init__(self, t,  ho, sha, ghAries0, ghAries1, dec, pitch=0):

    # the stuff in here, we just want to exist for THAT INSTANCE

    # everything that is an "input" to the object
    self.t = t
    self.ho = ho
    self.ghAries0 = ghAries0
    self.ghAries1 = ghAries1
    self.sha = self.convertAngle(sha)
    self.dec = self.convertAngle(dec)
    self.pitch = pitch

    # everything that requires a method to get
    self.increment = self.calcIncrement()
    self.hourTea = 0
    self.ghaA = self.interpGhaDec(self.ghAries0, self.ghAries1)
```

```
        self.gha = self.calcTrueGha()
        self.fixTime = t
        self.trackStar(self.lf, self.bf)

    def convertAngle(self, angle):
        # converts an angle from angle + minutes from the almanac to decimal
        # not gonna lie this looks whack.  --future self
        if angle[0] < 0:
            ang = -1*(abs(angle[0]) + angle[1]/60)
        else:
            ang = (abs(angle[0]) + angle[1]/60)
        return ang

    def calcIncrement(self):
        # increment is used to interpolate GHA. Not required until we're
        # automating this mofo

        increment = self.t[1]/60 + self.t[2]/3600
        return increment

    def calcHourTea(self):
        # need to convert everything to seconds
        hoursToSeconds = (self.t[0] - self.fixTime[0])*3600
        minsToSeconds = (self.t[1] - self.fixTime[1])*60
        secsToSeconds = (self.t[2] -self.fixTime[2])*1

        totalDiff = hoursToSeconds + minsToSeconds + secsToSeconds
        return totalDiff/3600       # need the final answer in HOURS
```

```
    def interpGhaDec(self, angle0, angle1):
        # attempts the interpolation method for getting GHA.
        # might make the automated look up table part a lot easier

        inc = self.increment
        qty0 = angle0
        qty1 = angle1

        #convert GHAs to decimal degrees
        a0 = (qty0[0] + (qty0[1]/60))
        a1 = (qty1[0] + (qty1[1]/60))

        # GHA = ghA0 + inc*(ghA1 -ghA0)
        output = (a0 + inc*(a1 - a0))
        return output

    def calcLongLat(self, longEst, latEst):
#        print "calcLongLat"
        # calculates the estimate of lat and long based on the running track
        # of the vessel

        lng = longEst + ((self.hourTea*(self.v/60)*sind(self.track))/cosd(latEst))
        lat = (latEst + (self.hourTea*(self.v/60)*cosd(self.track)))
        #print "lng = %s lat = %s" %(lng, lat)
        #print "pitch = %s" %self.pitch
        return [lng, lat]
    def calcLha(self, longEst):
        # calculates the local hour angle
        # lha = (gha + long)%360
```

```
        return (self.gha + longEst)#%360
    def calcHcAz(self,latEst):
        # calculates s, c, and altitude Hc
        s = sind(self.dec)
        c = cosd(self.dec)*cosd(self.lha)
        hc = asind(s*sind(latEst) + c*cosd(latEst))
        X = (s*cosd(latEst) - c*sind(latEst))/(cosd(hc))
        if X > 1:
            X = 1
        elif X < -1:
            X = -1
        else:
            pass
        A = acosd(X)

        # calculate the azimuth Z based on lha
        if self.lha > 180:
            Z = A
        else:
            Z = 360-A
        return [hc, Z]

    def calcIntercept(self):
        # calculates the intercept, which along with Z, and Hc are the most
        # important attributes
        return self.ho - self.hc

    def trackStar(self, longitude, latitude):
```

```
        self.lng = self.calcLongLat(longitude, latitude)[0]
        self.lat = self.calcLongLat(longitude, latitude)[1]
        self.lha = self.calcLha(longitude)
        self.hc = self.calcHcAz(latitude)[0]
        self.z = self.calcHcAz(latitude)[1]
        self.p = self.calcIntercept()


    def printAns(self):
        # prints the data for the different observations
        print "t == " + str(self.t)
        print "ho == " + str(self.ho)
        print "ghAries0 == " + str(self.ghAries0)
        print "ghAries1 == " + str(self.ghAries1)
        print "sha == " + str(self.sha)
        print "increment == " + str(self.increment)
        print "hourTea == " + str(self.hourTea)
        print "ghaA == " + str(self.ghaA)
        print "gha == " + str(self.gha)
        print "dec == " + str(self.dec)
        # print "count == " + str(self.count)
        print "lng == " + str(self.lng)
        print "lat == " + str(self.lat)
        print "lha == " + str(self.lha)
        print "hc == " + str(self.hc)
        print "z == " + str(self.z)
        print "p == " + str(self.p)
        print "pitch" + str(self.pitch)
```

```
def writeInertial(self, fileName):
    # this writes the inertial logfile for that observation
    fileName.write('t == %s\n' %(self.t))
    fileName.write('ho == %s\n' %(self.ho))
    fileName.write('ghAries0 == %s\n' %(self.ghAries0))
    fileName.write('ghAries1 == %s\n' %(self.ghAries1))
    fileName.write('sha == %s\n' %(self.sha))
    fileName.write('increment == %s\n' %(self.increment))
    fileName.write('hourTea == %s\n' %(self.hourTea))
    fileName.write('ghaA == %s\n' %(self.ghaA))
    fileName.write('gha == %s\n' %(self.gha))
    fileName.write('dec == %s\n' %(self.dec))
    fileName.write('\n\n\n')
    return self.ho


def writeChanging(self,fileName):

    fileName.write('lng == %s\n' %(self.lng))
    fileName.write('lat == %s\n' %(self.lat))
    fileName.write('lha == %s\n' %(self.lha))
    fileName.write('hc == %s\n' %(self.hc))
    fileName.write('z == %s\n' %(self.z))
    fileName.write('p == %s\n' %(self.p))
    fileName.write('\n\n\n')
    return self.hc

def writeAll(self, inertialList, changingList):
    for x in fileList:
```

```
            writeInertial(x)
            writeChanging(x)

    def writeDebug(self, fileName):
        inertials.write("%s,%s,%s\n"%(self.lha, self.gha, self.increment))

def locateMe(obs, count, newLong, newLat):
    # this function will locate me based on the three observations
    # calculates A-G
    # increments count on subsequent iterations


    # set up matrices
    azimuth = [] #
    intercepts = []
    A = []
    B = []
    C = []
    D = []
    E = []
    # G = AC - B^2

    # single value parameters
    if count == 0:
        #print "count == 0"
        oldLat = obs[0].bf
        oldLong = obs[0].lf
    elif count > 0:
        #print "count > 0"
```

```
    oldLat = newLat
    oldLong = newLong


# get azimuths from the set of observations
[azimuth.append(x.z) for x in obs]

# get intercepts from the set of observations
[intercepts.append(x.p) for x in obs]

# calculate parameters
[A.append(cosd(x)**2) for x in azimuth]
A = sum(A)
[B.append(cosd(x)*sind(x)) for x in azimuth]
B = sum(B)

[C.append(sind(x)**2) for x in azimuth]
C = sum(C)

[D.append(intercepts[i]*cosd(azimuth[i])) for i in range(0, len(azimuth))]
D = sum(D)

[E.append(intercepts[i]*sind(azimuth[i])) for i in range(0, len(azimuth))]
E = sum(E)

G = A*C - B**2


# print " a = %s" %str(A)
```

```
    # print " b = %s" %str(B)
    # print " c = %s" %str(C)
    # print " d = %s" %str(D)
    # print " e = %s" %str(E)
    # print " g = %s" %str(G)

    # FIRST PASS
    # calculate longitude from the averaging method
    newLongOffset = (A*E - B*D)/(G*cosd(oldLat))
    newLong = (oldLong + newLongOffset) #i'm trying mod 360
    #print "long off == %s " %str(newLongOffset)
    #print "newlong == %s " %str(newLong)

    # calculate latitude from the averaging method
    newLatOffset = (C*D-B*E)/(G)
    newLat  = (oldLat +newLatOffset)
    #print newLat
    #print "n lat off == %s" %str(newLatOffset)

    d = 60*m.sqrt((newLong-oldLong)**2*cosd(oldLat)**2 + (newLat - oldLat)**2)
#    print  "d == %s" %str(d)
    dLog.write("%s\n" %d)

    return [newLong, newLat, d]

def runLocateMeALot(obs):
inertialList = [inertialParams, ip1, ip2]
        changingList = [changingParams, cp1, cp2]
        # first time
```

```
count =0

location = locateMe(obs, count,0,0)
# print "location is %s" %str(location)
hoiList = []
hciList = []
angleDiff = []
for x in obs:
    hoi = x.writeInertial(inertialParams)
    hci = x.writeChanging(changingParams)
    hoiList.append(hoi)
    hciList.append(hci)
print "hoilist =%s"%hoiList
print "hciList=%s"%hciList

# iterate the algorithm 10x
while count < 10:
    for x in obs:
        x.trackStar(location[0], location[1])
#        print " new long = %s" %str(x.lng)
#        print "new lat = %s" %str(x.lat)
       # print "new lha = %s" %str(x.lha)
       # print "new hc = %s" %str(x.hc)
       # print "new z = %s" %str(x.z)
       # print "new p = %s" %str(x.p)
    count = count + 1
    location = locateMe(obs, count, location[0], location [1])
    for x in obs:
        x.writeInertial(inertialParams)
```

```
                x.writeChanging(changingParams)
                x.writeDebug(inertials)
            if count == 10:
                dLogFinalPoint.write("%s,%s,%s\n"
                %(location[0], location[1], location[2]))
                # put in the amount of pixels away the
                analysis.write("%s,%s,%s,%s,%s,%s,%s,%s,
                %s\n"%(hoiList[0],hoiList[1],hoiList[2],
                hciList[0],hciList[1],hciList[2],
                location[0],location[1],location[2]))
                return location

# if __name__ lets you run a code when its being run directly.
# this way is OP when you want to test a part of the code separately
if __name__ == "__main__":
    mcCount = 0
    while mcCount < 1000: # use 1000 for sims, 1 for obs
        # the monte carlo test has to start around this part of the code
        # these are the sample observations from the almanac
                            #t            Ho      sha          gha0
        noiseBound = 0#0.1666 # 0.00625
        randomNumber = rng.uniform(-noiseBound,noiseBound)
        randomNumber1 = rng.uniform(-noiseBound,noiseBound)
        randomNumber2 = rng.uniform(-noiseBound,noiseBound)
        bias =0#-2  #-15
        bias2 = 0


        # these observations are from the nautical almanac 2015
```

```
         #t          h0                                  sha            gha0
regulus = Observation([20, 39, 23], 27.2674 ,[207, 42.3], [222, 30.6],
                      #gha1        dec
                      [237, 33.1], [11, 53.4])

antares = Observation([20, 45, 47], 25.8742 , [112, 24.2], [222, 30.6],
                      [237, 33.1], [-26, 27.8])

kochab = Observation([21, 10, 34], 47.5309 , [137, 19.7], [237, 33.1],
                      [252, 35.6], [74, 5.9])



t = [0,53,13]
ghaZero = [16,18.7]
ghaOne = [31, 21.1]
meanBias =5.356 # total amount of diff. i need to get out of my system
dt = 0
uncertainty = 0.2222#0.1415
tBias = 1
r0 = rng.uniform(-uncertainty,uncertainty)
r1 = rng.uniform(-uncertainty,uncertainty)
r2 = rng.uniform(-uncertainty,uncertainty)
t1 = rng.uniform(-tBias, tBias)
t2 = rng.uniform(-tBias, tBias)
t3 = rng.uniform(-tBias, tBias)

# 3 stars sighted the same time
obs = [regulus, antares, kochab]
```

```
runLocateMeALot(obs)
        h = Sextant(29, 170)
        mcCount +=1
```

Appendix B

Center Pixel Method Script

This appendix contains the source code which computed the center pixel method in the rectangular and gnomonic coordinate system. It also runs the blind astrometric calibration software for image processing.

```
# rewriting the center pixel method for use in python
# eliminates the need to do any kind of "circular hough transform" etc.
# USES ASTROMETRY "lang et al"
# tckf october 2015



from __future__ import division
from PIL import Image


try:
    import client
except:
    pass
import time
import sys
```

```
import os
import subprocess
import json
# brittle code. need a way to do relative path
sys.path.append('/home/newmy/research/exp/unh
-startracker/unh-startracker/analysisAndPlots/maps/')
# print sys.path
import re
import glob
import ST

f = open('plotData.csv', 'w')
zz = open('imageList.csv','w')
'
dataDir = '/home/newmy/research/exp/unh-startracker/dataSets/timeVarying/'
#dataDir = './../'
print "ATTEMPTING TO ACCESS DATA DIR %s" %dataDir
def extractTimeFromImage(img):
    img = Image.open(img)
    imgTime = img._getexif()
    imgTime = imgTime[36868].split()
    imgTime = imgTime[1].split(':')

    # imgTime is an array of [hr, min, sec] in UTC from the image file
    imgTime =  [float(x) for x in imgTime]
    return imgTime

def populateImageList():
    imageList = glob.glob('%s*.jpg'%dataDir)
```

```
    print imageList
    return imageList

def grabWcsFile(image):
    # i think what i really want is to return a string of the wcs file
    filename = os.path.basename(image)
    filename = os.path.splitext(filename)
    filename = "%s.%s"%(filename[0], 'wcs')
    #print "filename === %s" %filename
    return filename

try:
    def apiGetCalibrationAndStarList(image):
        # UNUSED SINCE WE HAVE THE SOFTWARE LOCALLY NOW
        apiKey = 'lmrqojqbcjrzxkzx'
        c = client.Client()
        c.login(apiKey)

        # the reason we set equal c.upload to a variable is because it returns a
        # dictionary with the image id in it
        uploadedImage = c.upload(image)

        #print "ul image == %s" %uploadedImage
        # time.sleep(30)
        # try to minimize
        while True: # look forever
            subId =  uploadedImage['subid']
            if subId == None:
                time.sleep(5)
```

```
    elif subId != None:
        break
# print subId
while True:
#NEED TO ACTIVATE THE JUSTDICT = TRUE FLAG
    blah = c.sub_status(subId, justdict=True)
    try:
        # print "blah ========== %s" %type(blah['jobs'][0])
        if type(blah['jobs'][0]) == int:
            print "TYEP OF blah ========== %s" %type(blah['jobs'])
            jobId = blah['jobs'][0]
            print "jobId ============== %s" % jobId
            break
    except:
        time.sleep(5)
        pass


print "%s" % blah

# this time.sleep call will need to be more bulletproof
time.sleep(60)
calibration = c.send_request('jobs/%s/calibration' % jobId)
starList = c.send_request('jobs/%s/annotations' % jobId)
#while True:
#    try:
#        if calibration[''
#while True:
 #   calibration = c.send_request('jobs/%s/calibration' % jobId)
```

```
        #   starList = c.send_request('jobs/%s/annotations' % jobId)
        #   try:
        #       if exists(calibration['parity']):
        #           return [calibration, starList]
        #   except:
        #       time.sleep(5)
        #       pass


    # define center pixel constants, pitch calibration and stuff
except:
    pass

def makeWcsFiles(image):
    # this function calls astrometry with certain options and puts a .wcs file
    # in the directory specified after '-D'
    subprocess.call(['solve-field', '-p', '--overwrite', '--scale-units',
                     'degwidth', '--scale-low', '25', '--scale-high', '35',
                     '--downsample', '2', '-D', dataDir, image])

def getStarDict(wcsFile):
    # start a star dictionary object (kind of)
    wcsString = '%s%s'%(dataDir,wcsFile)
    command = ['/usr/local/astrometry/bin/plotann.py',
    '--brightcat',
    '/home/newmy/research/exp/unh-startracker/astrometry.net-0.50/
    catalogs/brightstars.fits',
     wcsString]
    print 'command is %s' %command
```

```
    sdProc = subprocess.Popen(command, stdout=subprocess.PIPE)
    #sdProc = os.popen(command).read()
    starDict = sdProc.stdout.read()

    # put the dictionary in dictionary (instead of string) form
    starDict = json.loads(starDict)
    print "star dictionary == %s"%type(starDict)
    print starDict
    return starDict

def computeHoFromImage(starList):

    # the leading theory is that this approximation will
    #only work when the stars are closeby
    # that is, the small angle approximation works
    pixScale = 24.1#5 # arcsec/pixel
    pixDegrees = pixScale/3600
    K = 3600/24.2
    imageWidth = 3872 # pix
    imageHeight = 2592 # pix
    xc = imageWidth/2
    yc = imageHeight/2
    #naturalBias = 0 # default value
    naturalBias = -5.0244 #KEEP THIS ONE. HYBRID OPTIMAL
    #PITCH AND ROLL FROM IMU CALIBRATION SUPER IMPORTANT
    pitch = 62.62 #39.76#62.62 #deg
    roll =-0.54#1.9 #-0.32-1.9056-3.19+4444#-0.5 # -2.4#-0.54
    #roll = -0.32
    rollOffset = 0 # default value
```

```
rollOffset = -2.3799 #KEEP THIS ONE. HYBRID OPTIMAL
#rollOffset = 26.0804 #gnomonic

shortList = [] # make a short list of the stars we're really interested in
for star in starList:
    # if the star has an alternate name its probably big or bright or both
    #if len(star['names'])>1:
        # pixel offsets

    # newx and newy are no longer used in the actual calculation
    star['newx'] =
     star['pixelx']*ST.cosd(roll+rollOffset) +
     star['pixely']*ST.sind(roll+rollOffset)
    star['newy'] =
    -star['pixelx']*ST.sind(roll+rollOffset) +
     star['pixely']*ST.cosd(roll+rollOffset)
    star['xOff'] = star['pixelx'] - xc
    star['yOff'] = yc - star['pixely']
    # rotation will go here
    # compute offset from roll angle
    # i think the order we do the roll and pitch operations is kind of important

    ####################################
    ####################################
    # THIS LINE IS THE RECTANGULAR TRANSLATION FOR THE STAR CAMERA
    star['ho'] =
     pixDegrees*star['xOff']*ST.sind(roll+rollOffset) +
     (pixDegrees*star['yOff'] + (pitch +
      naturalBias))*ST.cosd(roll+rollOffset)
```

```
        ####################################
        ####################################

        ####################################
        ####################################
        # THIS LINE IS THE GNOMONIC PROJECTION TRANSLATION
        # IF U WANT TO MESS WITH THE EQN, U CAN WRITE A NEW
        ####################################
        ####################################
        #star['ho']
         pixDegrees*star['xOff']*ST.sind(roll+rollOffset)+
        ST.gnomonicProjection(pixDegrees*star['xOff'],
        pixDegrees*star['yOff'],pitch+naturalBias)*ST.cosd(roll+rollOffset)
        #shortList.append(star)
        print star
        # print "\n\n\n shortList \n\n\n" %shortList
    return [starList, pitch, roll]

def calibrationObservation(shortList, t0, pitch, roll):
   # this function runs ST.py with the values from three
   #stars from the "short List"
    # we still need data from the almanac but there's
    #defintiely a way to leverage the astrometry data
    #t0[2] = t0[2] + 4 #lets worry about correcting the
    shorterList = []
    for x in range(0, len(shortList)):
        for name in shortList[x]['names']:
            #print shortList[x]['names'][name]
            print name
```

```
starName = name
if starName  == '14Aql':
    shortList[x]['sha'] = [74, 16.375]
    shortList[x]['dec'] = [-3,41.9393]
    shorterList.append(shortList[x])
elif starName == '15Aql':
    shortList[x]['sha'] = [73, 26.2652]
    shortList[x]['dec'] = [-4, 52.95334]

    shorterList.append(shortList[x])
elif starName == u'\u03bb Aql':
    shortList[x]['sha'] = [73.0, 26.26254]
    shortList[x]['dec'] = [-4,52.95335]
    shorterList.append(shortList[x])
elif starName  == 'Vega':
    shortList[x]['sha'] = [80, 47.02134]
    shortList[x]['dec'] = [38, 47.1234]
    shorterList.append(shortList[x])
elif starName == 'Sheliak':
    shortList[x]['sha'] = [77, 28.8012]
    shortList[x]['dec'] = [33, 21.7601]
    shorterList.append(shortList[x])
elif starName == 'Sulafat':
    shortList[x]['sha'] = [75, 15.8444]
    shortList[x]['dec'] = [32, 41.3734]
    shorterList.append(shortList[x])
else:
    pass
```

```
    print "actually used observations are ===== %s" %shorterList
    observations = []

    for x in shorterList:
        # have to define the gha as constants for now
        #gha0 = [17,17.8]
        #gha1 = [32, 20.3]
        gha0 = [32, 20.3]
        gha1 = [47, 33.7]
        Ho = x['ho']
        sha = x['sha']
        dec = x['dec']
        observations.append(ST.Observation(t0,Ho, sha, gha0, gha1, dec, pitch))
        f.write('%s,%s,%s,%s,%s,%s,%s,%s,%s\n'%(x['ho'],x['newx'], x['newy'],
        x['pixelx'], x['pixely'],x['xOff'],x['yOff'], pitch, roll))
    print "OBSERVATIONS ====== %s"%observations
    ST.runLocateMeALot(observations)

def runCenterPixelMethod():
    # what would go in __name __ == "__main__" so that way it can be easily
    # exported to another module if one day we need that
    imageList = populateImageList()
    zz.write(str(imageList)+"\n")
    for image in imageList:
        # produce the initial wcs file
        makeWcsFiles(image)
        t0 = extractTimeFromImage(image)
        wcsFile = grabWcsFile(image)
```

```
        print "wcsFile ========= %s " %wcsFile
        starList = getStarDict(wcsFile)
        temp = computeHoFromImage(starList)
        shortList = temp[0]
        pitch = temp[1]
        roll = temp[2]
        calibrationObservation(shortList, t0, pitch, roll)

    subprocess.call(['mv', 'analysis.csv', dataDir])
    subprocess.call(['mv', 'trueInertials.csv', dataDir])
    subprocess.call(['mv', 'plotData.csv', dataDir])



if __name__ == "__main__":
    f.write('ho,newx,newy,pixelx,pixely,xoff,yoff,pitch,roll\n')
    runCenterPixelMethod()
    f.close()
    zz.close()
```

Appendix C

Cost Function Optimization Script

This MATLAB script optimized over the rectangular and gnomonic cost function presented in Chapter 4.

```
% this script tries to plot the error in the intercept by using the actual
% cost function.
% tckf spring 2016
clear all; close all;
plotData = csvread('plotData.csv', 1, 0);
analysis = csvread('analysis.csv');
%conversion factor degrees to pixels
K = 3600/24.2; % the signs are basically opposite due to the weird origin
% pulling out variables that we need
x = plotData(:, 6);
y = plotData(:, 7);

%degrees originall. we mult by K to convert to pixels
pitch = mean(plotData(:,8))*K;
```

```
roll = mean(plotData(:,9));

dataPoints = 200
ranges = 360
% the roll and pitch offsets will go here
% remember these are possible errors.
rollOff = linspace(-ranges,ranges, dataPoints);
pitchOff = linspace(-ranges, ranges, dataPoints);

% roll and pitch operations, but idk the order yet
[PO, RO] = meshgrid(pitchOff, rollOff);

% the normalization happen here if desired.
%PO = PO;
%RO = RO;


%tht = roll;%+ RO;
% need to just do one computation but then repeat it a bunch of times.
newy = [];
newy1 = [];
newy2 = [];


y1= y(1:3:end);
y2 = y(2:3:end);
y3 = y(3:3:end);
```

```
HC1 = analysis(:,4);
HC2 = analysis(:,5);
HC3 = analysis(:,6);
HCT = [HC1;HC2;HC3];
DC1 = HC1 - pitch;
DC2 = HC2 - pitch;
DC3 = HC3 - pitch;

%convert Desired Center angles into desired center pixels
DP1 = DC1*K;
DP2 = DC2*K;
DP3 = DC3*K;
G1 = newy-DP1(1);
G2 = newy1-DP2(1);
G3 = newy2-DP3(1);
bound = length(x);
E = cell(bound,1);

% populate newy matrix based on rectangular
for n = 1:1:bound
for j = 1:length(pitchOff)
    for i = 1:length(rollOff)
        tht = roll + rollOff(i);
        newy(i,j) = (-x(n)*sind(tht) + (y(n) +(pitch+pitchOff(j)*K))*cosd(tht));%+;
    end
end
E{n} = (1/1)*((newy - HCT(n)*K).^2);
end
```

```
%populate newy matrix based on gnomonic projection
%for n = 1:1:bound
%for j = 1:length(pitchOff)
%    for i = 1:length(rollOff)
%        tht = roll + rollOff(i);
%        p = sqrt((-x(n))^2 + y(n)^2);
%        c = atan2(p,1);
%        phi1 = 0;
%        newy(i,j) = (x(n)*sind(tht) + (y(n) +pitch+K*pitchOff(j) +
%        K*atan2(x(n)*sin(c),p*cos(phi1)*cos(c)
-y(n)*sin(phi1)*sin(c)))*cosd(tht));

end
E{n} = (1/1)*((newy - HCT(n)*K).^2);
end
%add all the matrices E together
Etotal = zeros(dataPoints);
for i = 1:bound;
    Etotal = E{i} + Etotal;
end

% find optimal values for use in the star camera.
[r,c] = find(Etotal==min(min(Etotal)))
pitchOpt = rollOff(r)
rollOpt = pitchOff(c)
```