Optical Orbit Tracking and Estimation

Matthew Gillette
Embry-Riddle Aeronautical University, gilletm3@my.erau.edu

ACKNOWLEDGEMENTS

The author would like to thank the individuals who contributed to this work and made it possible, both directly and indirectly. First, I would like to thank my parents and grandparents for their support and encouragement, which has been a constant from day 1 as a freshman. To David Zuehlke, for giving me this idea and for pointing me in the right direction at a moment's notice when I needed it. To Dr. Henderson, whose instruction both inside and outside of class made it possible to piece together the research done here. To Dr. Prazenica, for helping me to realize, through his instruction both in and out of class, that it was not only possible for me to solve the problems put before me, but to understand the subject matter involved in doing so. Finally, thank you to Dr. Perera, whose patience and eagerness to help when learning became a struggle led to understanding linear algebra in a way that, at many times, seemed out of reach. The help of everyone mentioned here has been invaluable in ways that are both tangible and intangible, and there is simply no way to fully express my gratitude to each of them.

ABSTRACT

Angles-only initial orbit determination methods are currently limited in their use as they require some prior knowledge of where the observed object will be and when it will be there. This research aims to produce a viable method to automate this process so that objects whose trajectories are not saved in a user’s catalog can be observed. This study presents a novel approach to satellite recognition in an image. This method is used in addition to Astrometry to determine the right ascension and declination of the object. This information is then used to either obtain the initial conditions needed for a state estimator or is utilized by a Kalman Filter to correct any resulting error. In addition, an extra goal is set to create a modular process so that any stage of the end-to-end process can be changed to suit a user’s individual needs while still being able to perform the task for which it was assigned. Tests with varying times between measurements were first run to determine if a discrete-time Kalman Filter is a viable method to correct the error created by the state estimator, where coordinates were fed directly into the filter with no images. The results of these tests demonstrated the successful application of the Kalman Filter in adjusting the projections made by the mathematical projections of the satellite’s trajectory based on the measured data. After this, tests were done on images that were acquired in a manner similar to how the filter would have acquired them to test the entire end-to-end process. This test results signify that the efficacy of the Kalman Filter in aligning the mathematical projections of the satellite’s trajectory with the measurement data, thus demonstrating its successful application. This means that if the projection is in the middle of the image, the centroid of the satellite streak will be in the middle well. A final test was conducted to implement the Kalman Filter on a telescope system while utilizing a different image processing technique. This test demonstrated that the Kalman Filter worked as intended in real time.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

TABLE OF CONTENTS iii

LIST OF FIGURES vi

1 Introduction 1

1.1 Problem Statement 1

1.2 Motivation 1

1.3 Summary 2

1.3.1 Research Contributions 2

2 Review of the Relevant Literature 3

2.1 State-Space Representation 3

2.2 State-Feedback Control 4

2.3 State Estimation and Kalman Filtering 5

2.4 Computing the Kalman Gain 6

2.5 Duality 7

2.6 Discrete-Time Systems and the Kalman Filter 9

2.7 Coordinate Systems 10

2.8 Image processing 11

2.8.1 Noise Reduction 11

2.8.2 Noise Filtering 12
2.8.3 Object Detection 12
2.9 Cost Reduction 15
2.9.1 Aitken’s and Neville’s Algorithm For Interpolation 15
2.9.2 Singular Value Decomposition 15
2.10 Hypothesis 17
3 Methodology 18
3.1 Procedure 18
3.1.1 Matrix Setup 18
3.1.2 Image processing and Initial Satellite Recognition 21
3.1.3 The Kalman Filter 26
3.2 Software 27
3.3 Hardware 28
3.4 Images and Testing Data 29
4 Results 31
4.1 Image Testing 31
4.2 Reliability Testing 34
4.3 Hypothesis Testing 40
4.4 Image Testing 44
5 Discussions, Conclusions, and Recommendations 47
5.1 Discussion 47

5.2 Future Work 48

5.2.1 Image Processing and Satellite Recognition 49

5.2.2 Changes to the Dynamics 49

5.3 Conclusions 51

6 REFERENCES 53

LIST OF FIGURES

Figure Page

Figure 3.1 The RA over time for 6 GEO satellites. 19

Figure 3.2 The DEC over time for 6 GEO satellites. 19

Figure 3.3 (Left) Image after the Hough Transform. (Right) Close-up of satellite. 23

Figure 3.4 The resultant line after taking the mean of all Hough lines. 24

Figure 3.5 The propagation lines that result from the first image. 24

Figure 3.6 The discrete-time Kalman Filter. 27

Figure 4.1 (Left) Result of Process 1. (Right) Result of Process 2. 32

Figure 4.2 Close-up of one line from Process 1 that falls along 3 apparent stars. 32

Figure 4.3 (Left) Process 1 on poorly filtered image. (Right) Process 2 on same image. 33

Figure 4.4 (Left) Lines from Process 1 after threshold increase. (Right) Lines from Process 2 for the same image. 33

Figure 4.5 RA for the continuous-time test. 34

Figure 4.6 DEC for the continuous-time test. 35

Figure 4.7 Error for the continuous-time test. 35

Figure 4.8 RA for typical Δt of 5.5 seconds. 36

Figure 4.9 DEC for typical Δt of 5.5 seconds. 36

Figure 4.10 Propagation error for typical Δt of 5.5 seconds. 37

Figure 4.11 Covariance for propagation step for typical Δt of 5.5 seconds. 37

Figure 4.12 Covariance for update step for typical Δt of 5.5 seconds. 37

Figure 4.13 RA for a typical Δt of 1 minute. 38

Figure 4.14 DEC for a typical Δt of 1 minute. 38

Figure 4.15 Propagation error for a typical Δt of 1 minute. 39

Figure 4.16 Covariance for propagation step for typical Δt of 1 minute. 39

Figure 4.17 Covariance for update step for typical Δt of 1 minute. 39

Figure 4.18 End-to-end RA for typical Δt of 10 seconds. 40

Figure 4.19 End-to-end DEC for typical Δt of 10 seconds. 40

Figure 4.20 End-to-end error for typical Δt of 10 seconds. 41

Figure 4.21 End-to-end covariance for the propagation step for typical Δt of 10 seconds. 41

Figure 4.22 End-to-end covariance for the propagation step for typical Δt of 10 seconds. 42

Figure 4.23 End-to-end RA for varying Δt of 40, 50, or 76 seconds. 43

Figure 4.24 End-to-end DEC for varying Δt of 40, 50, or 76 seconds. 43

Figure 4.25 End-to-end error for varying Δt of 40, 50, or 76 seconds. 43

Figure 4.26 End-to-end covariance for propagation step for varying Δt of 40, 50, or 76 seconds. 44

Figure 4.27 End-to-end covariance for the update step with varying Δt of 40, 50, or 76 seconds. 44

Figure 4.28 Right Ascension over time for the implementation test. 45

Figure 4.29 The declination over time for the implementation test. 45

Figure 4.30 The error over time for the implementation test. 46

Figure 4.31 Propagated covariance for the implementation test. 46

Figure 4.32 Updated covariance for the implementation test. 46

1 Introduction

The problem of determining the orbit of a body beyond Earth's atmosphere has long been on the minds of mathematicians and astronomers. Historically, this centered on the motion of planets and stars. Now, the focus is on objects much closer such as the ever-increasing number of satellites and other objects in orbit around the Earth. In recent years, research has been done to utilize angles-only initial orbit determination (AIOD) methods for objects in orbit around the Earth, primarily satellites. For more information on AIOD, its history, and the current challenge of monitoring the expanding number of objects in the sky, see Refs. [1,2]

1.1 Problem Statement

To date, the only way to get images across the sky of an object to use for an AIOD method is to utilize a software program or other means of knowing where the object will be and when it will be there. From here, the coordinates and the time for the image to be taken are then input into the software that interfaces with the telescope mount and camera. The goal of this research is to meet this need with a state estimator with a Kalman Filter that is capable of projecting a satellite's path across the sky while making necessary adjustments after measurements are taken, all without a priori knowledge of the satellite's orbit.

1.2 Motivation

One of the main goals of using AIOD is that it should be able to do its work autonomously. As stated above, current methods of AIOD are not autonomous. Therefore, the only objects that can be observed are objects whose orbit is already known. This is a great aid with testing AIOD methods but eliminates it as a method for observing objects whose trajectory is not in the catalogue that is used. The answer may seem obvious to use a state estimator. However, a state estimator will inherently produce error because of factors such as noise in the measurements, noise in the physical process being done, and any mathematical errors present in the model itself. Therefore, the aim of this research is to combine these elements to create an end-to-end process that will get the images necessary to perform an AIOD process to estimate the orbit of an unknown object.

1.3 Summary

This research advances the existing efforts towards establishing a comprehensive AIOD process by developing an end-to-end methodology that encompasses image acquisition for precise orbit determination. This is accomplished through the creation of an algorithm for recognizing satellites in an initial image followed by a small propagation. This is done until enough initial conditions are gathered to begin using a state estimator. This state estimator will utilize a discrete-time Kalman-Filter to allow the estimator to, essentially, correct itself of any error that results from a variety of sources such as process noise, measurement noise, and any error inherent in the model.

1.3.1 Research Contributions

This research provides 3 key contributions. The main contribution is the use of a state estimator with a Kalman Filter to project the path of a satellite across the sky without knowing its orbit. Another contribution is a satellite recognition method based on the Hough Transformation. The final contribution is a short-term projection method used prior to when the filter begins operating. This method simply projects a line made from the results of the Hough Transformation based on a ratio of the time between the start of the original image and a subsequent image and the exposure time.

2 Review of the Relevant Literature

In this section, topics that are related to this research are explored, with a focus on the Kalman Filter and the two ways it can be applied, continuous-time and discrete-time. Other topics include coordinate systems, image processing, and some relevant methods from linear algebra focused on mathematical cost reduction.

2.1 State-Space Representation

Equation (2.1) shows a general form of an nth order differential equation, where a_0, a_1, …, a_n and b_0 denote constants, x is a function of time, and u denotes an input. Brogan [3] shows this can be converted into a series of first-order differential equations. To achieve this, the system can be redefined using a new set of variables with the intent of creating these first-order differential equations as shown in Equation (2.2). Taking the derivatives of these yields Equation (2.3).

$$\frac{d^n x}{dt^n} + a_{n-1}\frac{d^{n-1}x}{dt^{n-1}} + \cdots + a_1\frac{dx}{dt} + a_0 x = b_0 u \tag{2.1}$$

$$x_1 = x, x_2 = \frac{dx}{dt}, \ldots, x_n = \frac{d^{n-1}x}{dt^{n-1}} \tag{2.2}$$

$$\dot{x}_1 = \frac{dx}{dt}, \dot{x}_2 = \frac{d^2x}{dt^2}, \ldots, \dot{x}_n = \frac{d^n x}{dt^n} \tag{2.3}$$

If the system is linear, the original differential equation (in state-space form) can be reformatted into the form shown in Equation (2.4), where a general form for A, shown in Equation (2.5), is comprised of the coefficients that are multiplied by the state variables, and B is the coefficients that are multiplied by the inputs. Here, $\underline{x}$ is the state vector comprised of the states from Equation (2.2), while $\underline{u}$ is a vector containing the inputs.

$$\underline{\dot{x}} = A\underline{x} + B\underline{u} \tag{2.4}$$

$$A = \begin{bmatrix} \underline{0} & & I \\ -a_0 & \ldots & -a_{n-1} \end{bmatrix} \tag{2.5}$$

When trying to account for a second, also linear, differential equation at the same time as the first, the A matrices for each can be combined into a single A matrix using the format in Equation (2.6) where the states for the second equation start at n+1. This assumes that the two equations are independent of each other. If the states for each of the differential equations are linked (some of the states appear in both equations), the only change will be to one line of one or both of the zero matrices in the corners. This pattern is repeated for each additional differential equation that is added to this A matrix, where the A matrix for the individual equation falls on the diagonal.

$$A = \begin{bmatrix} A_1 & [0] \\ [0] & A_2 \end{bmatrix} \tag{2.6}$$

The B matrix is related to the inputs. The number of columns is determined by the number of inputs and the number of rows is determined by the number of states. The constants inside the matrix correspond to the constants the inputs are multiplied by in the differential equations.

Equation (2.7) shows the equation for the outputs, $\underline{y}$. These outputs come from the sensors that are used to detect either the states or the inputs. The matrices for the state sensors, C, and the input sensors, D, are formed based on the number of outputs. The simplest way to form these 2 matrices is to put a 1 corresponding to the position that the output for that row represents if the sensors are directly sensing the states or inputs.

$$\underline{y} = C\underline{x} + D\underline{u} \tag{2.7}$$

2.2 State-Feedback Control

The systems discussed in the previous section are the open-loop versions of those systems, meaning the control is designed around goals and any prior knowledge of the system and does not adjust itself based on the outputs. A closed-loop system, on the other hand, adjusts the input using information about the system's current states. This can be done using the states directly or by using sensor outputs. Using the states directly to control the system is called state-feedback control.

The state-feedback controller is purely theoretical because the sensor outputs, $\underline{y}$, are the only accessible signals that can be used for determining the current state. However, the creation of a state-feedback controller does involve methodology that will be useful in later sections. First, it must be determined whether the system is controllable. This is done by checking the rank of the control matrix, shown in Equation (2.8), which is explained in more detail in Ref. [4]. If the system is controllable, this means the eigenvalues of the controlled system can be arbitrarily placed. To do this, assume the control law in Equation (2.9) and substitute into the state-space form shown in Equation (2.4). This simplifies to Equation (2.10). From here, desired eigenvalues can be selected and applied by using Equation (2.11), where $\overline{\lambda}$ are the desired eigenvalues and λ are the actual eigenvalues and solving for the individual control gains in K [5].

$$P_c = [B \; AB \; A^2B \; \dots \; A^{n-1}B] \tag{2.8}$$

$$u = -K\underline{x} \tag{2.9}$$

$$\underline{\dot{x}} = (A - BK)\underline{x} \tag{2.10}$$

$$\det(A - BK) = (\lambda - \overline{\lambda}_1)(\lambda - \overline{\lambda}_2) \dots (\lambda - \overline{\lambda}_n) \tag{2.11}$$

2.3 State Estimation and Kalman Filtering

Since the states themselves cannot be directly used to control a system, one method to try to address this is to estimate the states and determine the control input needed from those estimates while also comparing to the sensor outputs. Estimation is only possible for an observable system. This means that the rank of the observability matrix, shown in Equation (2.12), is equal to the number of states. To accomplish this, a new set of states is created, shown in Equations (2.13) and

(2.14), where $\underline{\hat{x}}$ denotes the estimated states. State estimation will inherently produce error. Sometimes, this error can be accounted for. One example of this is using a Kalman filter to account for noise. To do this, a Kalman gain, L, is used to make the eigenvalues of A-LC stable, which will drive the error to zero over time. The more negative the eigenvalues, the faster convergence will occur. However, making the values in the Kalman gain too large may result in amplifying the sensor noise.

$$P_o = [C^T \ (CA)^T \ (CA^2)^T \dots (CA^{n-1})^T]^T \tag{2.12}$$

$$\dot{\underline{\hat{x}}} = A\underline{\hat{x}} + B\underline{u} \tag{2.13}$$

$$\underline{\hat{y}} = C\underline{\hat{x}} + D\underline{u} \tag{2.14}$$

Equation (2.15) shows the error between the actual and estimated states. This can be expanded to Equation (2.17) by substituting Equation (2.4) for $\dot{\underline{x}}$ and Equation (2.13) for $\dot{\underline{\hat{x}}}$ and simplifying. This can be further simplified by substituting Equations (2.7) and (2.14) for $\underline{y}$ and $\underline{\hat{y}}$ respectively to get Equation (2.18).

$$\underline{e} = \underline{x} - \underline{\hat{x}} \tag{2.15}$$

$$\dot{\underline{e}} = \dot{\underline{x}} - \dot{\underline{\hat{x}}} \tag{2.16}$$

$$\dot{\underline{e}} = A(\underline{x} - \underline{\hat{x}}) - L(\underline{y} - \underline{\hat{y}}) \tag{2.17}$$

$$\dot{\underline{e}} = (A - LC)\underline{e} \tag{2.18}$$

This result is important because stable eigenvalues for the matrix *A-LC* mean that the estimation error will go to zero over time.

2.4 Computing the Kalman Gain

One way to design the Kalman Gain is through eigenvalue placement. If a system is observable, then the eigenvalues of the matrix *A-LC* can be placed arbitrarily. As shown in Equation (2.19), this process is similar to the process described in Section 2.2. The main difference between here and Section 2.2 is solving for the individual constants in L instead of K.

$$\det(A - LC) = (\lambda - \bar{\lambda}_1)(\lambda - \bar{\lambda}_2) \ldots (\lambda - \bar{\lambda}_n) \tag{2.19}$$

If, at a minimum, the statistics of the sensor noise are known or can be reasonably approximated, it is possible to calculate the Kalman Gain with that information. Equation (2.20) shows a variation of the state-space system shown earlier with an added term that accounts for noise, where $\underline{v}$ is the process noise and G is used to map the process noise into the dynamics model. Equation (2.21) shows the sensor output for this system where $\underline{w}$ is the sensor noise.

$$\underline{\dot{x}} = A\underline{x} + B\underline{u} + G\underline{v} \tag{2.20}$$

$$\underline{y} = C\underline{x} + D\underline{u} + \underline{w} \tag{2.21}$$

The Kalman Gain is computed using Equation (2.22), where $\hat{R}$ represents the covariance of the measurement noise and P_∞, the steady state value of the estimation error covariance, is the solution to the Algebraic Riccati Equation shown in Equation (2.23) and must be a positive definite matrix. More information on this variation will come in Section 2.5. For this equation, $\hat{Q}$ is the process noise covariance [5].

$$L = P_\infty C^T \hat{R}^{-1} \tag{2.22}$$

$$0 = A^T P_\infty + P_\infty A - P_\infty C^T \hat{R}^{-1} C P_\infty + G\hat{Q}G \tag{2.23}$$

2.5 Duality

Another method to create a Kalman filter is to create a dual system and solve for its control gain. As Ogata demonstrates in [6], if a system is observable, this means that the eigenvalues for the observer can be placed arbitrarily. Since this is, in essence, the same as arbitrarily placing the eigenvalues for a state-feedback controller, the same methods can be used to solve for the Kalman filter. This property, known as duality, allows for the system to be redefined using the following:

$$A^T = \tilde{A} \tag{2.24}$$

$$B^T = \tilde{C} \tag{2.25}$$

$$C^T = \tilde{B} \tag{2.26}$$

$$D = \tilde{D} \tag{2.27}$$

$$L^T = \tilde{K} \tag{2.28}$$

This yields the new set of dual state-space equations shown in Equations (2.29) and (2.30). An important note about this new system is that the controllability matrix of the dual system is equal to the transpose of the observability matrix of the original system. This allows for the eigenvalues of the dual system to be arbitrarily placed. Using these new state space equations along with Equation (2.31), it can be shown that the eigenvalues of the controlled dual system are equal to the transpose of the eigenvalues of the state observer shown above. Therefore, designing the controller gains of the dual system is the same as designing the Kalman filter for the state observer of the real system.

$$\underline{\dot{\tilde{x}}} = \tilde{A}\underline{\tilde{x}} + \tilde{B}\underline{\tilde{u}} \tag{2.29}$$

$$\underline{\tilde{y}} = \tilde{C}\underline{\tilde{x}} + \tilde{D}\underline{\tilde{u}} \tag{2.30}$$

$$\underline{\tilde{u}} = -\tilde{K}\underline{\tilde{x}} \tag{2.31}$$

These results are important because they allow the use of MATLAB's *lqr* function to create the Kalman Gain. The function is shown in Equation (2.32) for how the function is normally used for determining control gains, where Q is a weighting matrix for the states and R is a weighting matrix for the control inputs. Equation (2.33) shows how to input the dual system which, after transposing $\tilde{K}$, will give the Kalman Gain.

$$[K, S_\infty, \lambda] = lqr(A, B, Q, R) \tag{2.32}$$

$$\left[\tilde{K}, \tilde{S}_\infty, \lambda\right] = lqr(\tilde{A}, \tilde{B}, \tilde{Q}, \tilde{R}) \tag{2.33}$$

This result comes from expanding on the dual system. Starting with the Algebraic Riccati Equation for the real system in Equation (2.34) [6], it can be seen that the same for the dual system will appear as shown in Equation (2.35). If it is assumed that Equations (2.36), (2.37), and (2.38) are true, then it can be seen that Equations (2.35) and (2.23) are, in fact, the same. This is what allows the use of MATLAB's lqr function to determine the Kalman Gain

$$0 = S_\infty A + A^T S_\infty - S_\infty B R^{-1} B^T S_\infty + Q \tag{2.34}$$

$$0 = \tilde{S}_\infty \tilde{A} + \tilde{A}^T \tilde{S}_\infty - \tilde{S}_\infty \tilde{B} \tilde{R}^{-1} \tilde{B}^T \tilde{S}_\infty + \tilde{Q} \tag{2.35}$$

$$\tilde{S}_\infty = P_\infty \tag{2.36}$$

$$\tilde{R} = \hat{R} \tag{2.37}$$

$$\tilde{Q} = G\hat{Q}G \tag{2.38}$$

2.6 Discrete-Time Systems and the Kalman Filter

The above solutions for the Kalman Filter are for a continuous-time system, meaning the sensors are taking readings at a fast enough rate that it can be assumed they are continuous. However, that is sometimes not the case. When sensor readings are spaced out over a period of several seconds and beyond, it is advisable to use a discrete-time Kalman Filter instead. To do this, first the continuous-time system must be converted to a discrete-time system as shown in Equations (2.39) to (2.41). The idea is to propagate by steps of size Δt instead of integrating continuously across that same time interval. [7]

$$F = e^{A\Delta t} \tag{2.39}$$

$$G = B \tag{2.40}$$

$$H = C \tag{2.41}$$

The process for determining the Kalman Filter using this method is shown in Equations (2.42) through (2.46). In these equations, $\underline{\hat{x}}_k^-$ and P_k^- comprise the "propagate" stage, where the model is propagated forward to generate the estimate. The last 2 equations to solve for $\underline{\hat{x}}_k^+$ and P_k^+ are the update stage, which use the Kalman Gain, L_k, to correct for the error between the estimate and the measurement. In these equations, P_k denotes the covariance matrix and $\underline{\tilde{y}}_k$ denotes the measurements.

$$\underline{\hat{x}}_k^- = F_{k-1}\underline{\hat{x}}_{k-1}^+ + G_{k-1}\underline{u}_{k-1} \tag{2.42}$$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{2.43}$$

$$L_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.44}$$

$$\underline{\hat{x}}_k^+ = \underline{\hat{x}}_k^- + L_k(\underline{\tilde{y}}_k - H_k\underline{\hat{x}}_k^-) \tag{2.45}$$

$$P_k^+ = (I - L_kH_k)P_k^-(I - L_kH_k)^T + L_kR_kL_k^T \tag{2.46}$$

The Kalman Gain must be recalculated after each propagation. This is because the covariance matrix will be different each iteration as the error between the estimated results and measurements changes. [8]

2.7 Coordinate Systems

The equatorial coordinate system provides a consistent way to represent the location of an object in the sky because the coordinates of the object are the same, at any given point in time, no matter where it is viewed from. The equatorial coordinate system is comprised of the object's right ascension and its declination angles. These coordinates happen to coincide with the latitude and longitude coordinates that are used to describe locations on the Earth, with the right ascension following along with longitude and declination with latitude. However, there is one key difference between right ascension and longitude. This difference is that right ascension divides the globe into 24 equal sections (hours) to represent one rotation of the Earth for a day, with a positive value (or hour angle) being to the east up until the 12-hour mark. West will result in a negative value. The location of the 0-hour mark is at the vernal equinox, the longitudinal position where the sun crosses the equator during its movement relative to the background stars. Despite this, it is simple to switch from hours to degrees since 1 hour is equal to 15 degrees [9].

2.8 Image processing

A digital image can be thought of as an intensity matrix, $I(x, y)$, where x and y are a pixel's coordinates and the number contained in each cell is a measurement of the photons collected at that location. This means that dimmer objects will be harder to differentiate from background objects. This can be remedied by increasing the exposure time for the image, as that will allow extra time for more photons to be collected by each pixel. However, this increases the noise in the image. Therefore, the goal of image processing here is to reduce the noise in an image so that features may become more prominent [10, p. 50].

2.8.1 Noise Reduction

There are several methods that can be employed to reduce the noise, some in conjunction with each other. One method is to use dark frames. A dark frame is an image that uses the same properties of a normal image, such as exposure time, with the telescope lens covered. This creates an image that captures some of the issues that will plague the normal image such as thermal noise and dead pixels. In addition, flat frames can be used to help correct issues with illumination in an image. This works by covering the camera's field of view with an evenly lit surface. The image is taken, keeping all settings the same as the normal image except for the exposure time. The exposure time is adjusted to make use of as much of the camera's dynamic range as possible. Flat frames help with correcting for issues such as areas that are darker. Flat frames may also show dust spots. The final frame type to help with noise is called a bias frame. A bias frame is similar to a dark frame with the exception that the exposure time is set to be as short as possible. This helps to account for electronic read noise and any damaged pixels that may affect an image [10, pp. 53-55].

2.8.2 Noise Filtering

Another approach to reducing the noise in the image, which can be done in addition to the reduction techniques discussed above, is to filter it. One way to do this is known as median filtering. Median filtering takes a small area of pixels and finds a median. If there is a significant change over a small area of pixels, there is likely noise in that area. A Gaussian smoothing filter attempts to suppress high-frequency content in an image by taking the convolution of a Gaussian kernel operator, shown in Equation (2.47) where σ is a chosen standard deviation, and creates a smoothed image where each pixel's value is the weighted average of the neighboring pixels.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{2.47}$$

The final filtering method is threshold filtering, which removes noise by determining if the intensity in a pixel reaches a certain threshold for a smoothed image. If the intensity is below the threshold, the pixel at that coordinate is set to 0. The threshold is arbitrarily set, which means it can be chosen to be at or above the mean intensity value for an image [10, pp. 55-57].

2.8.3 Object Detection

Centroiding, the practice of differentiating between objects and background and finding the approximate center of that object, is used to locate objects within an image so that they may be processed as noise, stars, or satellites. Noise and some smaller (relative to the image) stars can be filtered out by introducing a minimum area that an object must cover. One method that can be used, and is used in this research, is to use a weighted average of the intensities of the object as shown in Equation (2.48), where $\underline{x}_i$ is the coordinates of the i^{th} pixel, w_i is the intensity of the pixel, N is the number of pixels in the object, and $\underline{x}_c$ is the location of the centroid. Centroiding by way of intensity-weighting is capable of sub-pixel accuracy. In MATLAB, the function *regionprops* can be used to perform this task.

$$\underline{x}_c = \frac{\sum_{i=1}^{N} \underline{x}_i w_i}{\sum_{i=1}^{N} w_i} \tag{2.48}$$

The next step is to differentiate between point objects and streaks. For this, detecting corners is a useful way to find the streaks in an image. In a filtered image, there will be a strong bi-directional gradient at a corner, or a sharp change in the intensity [11]. A corner can be discovered using the eigenvalues of the structural tensor shown in Equation (2.49). If the minimum of the 2 eigenvalues of each 2x2 element of Z(x,y) exceeds a chosen threshold, then a corner has been detected. This is called the *minimum eigenvalue method*. However, this method will quickly become computationally expensive. One way to reduce the computational cost is to only search areas known to have objects from the above centroid detection. Another method is to use an approximation of the cornerness metric instead of solving for the eigenvalues outright. Harris Corner Detection uses a cornerness metric approximation, shown in Equation (2.50), to determine the likelihood that a particular pixel is a corner. A streak can be found by analyzing the distance between the corners as well as ensuring the respective corners correspond to a detected object. In MATLAB, the function detectHarrisFeatures can be used to accomplish this task.

$$Z(x,y) = \begin{bmatrix} \left(\frac{\delta^2 I(x,y)}{\delta x^2}\right)^2 & \frac{\delta I(x,y)}{\delta x}\frac{\delta I(x,y)}{\delta y} \\ \frac{\delta I(x,y)}{\delta x}\frac{\delta I(x,y)}{\delta y} & \left(\frac{\delta^2 I(x,y)}{\delta y^2}\right)^2 \end{bmatrix} \tag{2.49}$$

$$c_h(x,y) = \left(\frac{\delta^2 I(x,y)}{\delta x^2} * \frac{\delta^2 I(x,y)}{\delta y^2}\right)^2 - \left(\frac{\delta I(x,y)}{\delta x} * \frac{\delta I(x,y)}{\delta y}\right)^2 - k\left(\frac{\delta^2 I(x,y)}{\delta x^2} + \frac{\delta^2 I(x,y)}{\delta y^2}\right)^2 \tag{2.50}$$

To further reduce the computational cost of this method, the derivatives in Equations (2.49) and (2.50), known as the spatial derivatives, can be approximated using Prewitt operators shown in Equations (2.51) and (2.52). These can approximate a spatial derivative by simply multiplying them by the intensity matrix as shown in Equations (2.53) and (2.54). To get the second spatial derivative, simply multiply the corresponding Prewitt operator by the first spatial derivative.

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.51}$$

$$K_y = K_x^T \tag{2.52}$$

$$\frac{\delta I(x,y)}{\delta x} \approx K_x * I(x,y) \tag{2.53}$$

$$\frac{\delta I(x,y)}{\delta y} \approx K_y * I(x,y) \tag{2.54}$$

Any noise remaining in an image after reduction and filtering (Sections 2.6.1 and 2.6.2) may create issues with these results. The effects of the noise can be mitigated by smoothing the image using a Gaussian kernel shown in Equation (2.55), where $G(x,y,\sigma)$ comes from Equation (2.47).

$$I_{smooth} = G(x,y,\sigma) * I(x,y) \tag{2.55}$$

The Hough Transform can also be used to find streaks in an image by using parametric representations of line segments to locate the line segments. The Hough Transform uses thresholding parameters, such as streak length, to reduce the number of points in the image and create candidate segments from this reduced list. Adjusting these parameters will affect how many segments are detected [10, pp. 58-66].

2.9 Cost Reduction

A limiting factor to the number of images that can be taken during a satellite pass is the computation time required for an image to be processed and the states to be propagated. Therefore, effort should be put into attempting to reduce the computational complexity of any algorithm used to reduce the amount of time that elapses between images.

2.9.1 Aitken's and Neville's Algorithm For Interpolation

One method that can be used for interpolating points is called Aitken's and Neville's Algorithm. The algorithm, shown in Equation (2.56) uses a lower triangular matrix and forward elimination to create a polynomial that describes the series of points that are given, which is shown in Equation (2.57).

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & 0 & \cdots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_1)(x_1 - x_0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & (x_n - x_1)(x_1 - x_0) & \cdots & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.56}$$

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = y \tag{2.57}$$

The constants, denoted as a_i, are solved for by multiplying the vector of constants by the matrix, where each x_i has a corresponding solution y_i and setting each of those results equal to the corresponding row in the vector of solutions. The constants can then be solved for in the order they are presented in the vector [12].

2.9.2 Singular Value Decomposition

Singular value decomposition (SVD) is a linear algebra technique that involves breaking down a matrix into 3 matrices that, when multiplied together, equal the original matrix. This is shown in Equation (2.58), where A is an mxn matrix, U is an mxm orthogonal matrix whose columns are the eigenvectors of AA^T, Σ is an mxn diagonal matrix of singular values, and V is an nxn orthogonal matrix whose columns are the eigenvectors of A^TA. For the scope of this research, it is assumed that A is in the real domain, which means U, Σ, and V will also be in the real domain.

$$A = U\Sigma V^T \tag{2.58}$$

The off-diagonal terms of Σ will all be 0. The diagonal terms are equal to the square root of the eigenvalues of A^TA and are all positive. The order they are arranged in is decreasing magnitude so that the first value, the one in the top-left corner of the matrix, has the greatest value. The rank of A is equal to the number of nonzero elements in Σ. The terms on the diagonal of Σ are called the singular values of A. Since the singular values are arranged based on decreasing magnitude, Σ can be arranged as shown in Equation (2.59), where Σ_1 is a matrix of the nonzero singular values, arranged in descending order.

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.59}$$

This leads to arranging U and V as shown in Equations (2.61) and (2.62). Here, U_1 and V_1 are comprised of the eigenvectors that correspond to the nonzero eigenvalues of AA^T and A^TA respectively.

$$U = [U_1|U_2] \tag{2.60}$$

$$V = [V_1|V_2] \tag{2.62}$$

This presents an opportunity to create a compact version of A using Equation (2.62). The columns of U_1 and V_1 then form an orthonormal basis for the row space of A and A^T respectively. U_2 and V_2, on the other hand, form and orthonormal basis for the null space of A^T and A respectively [13].

$$A = U_1\Sigma_1 V_1 \tag{2.62}$$

An application of SVD is in corner or edge detection. This is done by taking a target pixel and then forming a matrix out of it and the 8 pixels immediately surrounding it. The SVD of this matrix is taken and then replace the original pixels with the maximum singular value and the gradient taken using a series of 8 convolution matrices to get a gradient set. This gradient set is compared to a threshold and, if the value for the pixel exceeds this threshold, it is a corner. If it does not exceed this threshold, it is not a corner [14].

SVD can also be used to eliminate noise in an image. This is because, after decomposing the image, noise is typically associated with the smaller singular values. Discarding these will result in a compressed matrix, as shown above [15]. In terms of its potential application here, using SVD for image processing is a topic for future research.

2.10 Hypothesis

The belief is that a Kalman Filter, when combined with a linear state estimator, will be able to project the path of a satellite in the RA-DEC coordinate frame while correcting error between projection and measurement.

3 Methodology

Now that the necessary mathematical background has been explored, these tools can now be combined to create a novel approach to the problem of automating the AIOD process. First the end-to-end process will be discussed in detail, which will include pointing out which parts are interchangeable so as to meet the adaptability goal of this research. After this, the software that is utilized in this research and the hardware setup used for acquiring the images are also discussed.

3.1 Procedure

This process will follow a format of first setting up the matrices that will remain constant throughout as well as proving that the designed matrices for A and C will allow the chosen methods to be used. This is followed by the image processing and satellite recognition techniques that were used in this research. These methods were used as a continuation of the research done by Zuehlke [10]. Finally, an explanation of how the Kalman Filter is provided. It is important to remember that the focus of this research is the Kalman Filter. Its process is designed to only need initial conditions. The method used to get these initial conditions is up to the user provided their chosen method elicits the required material. There are also several aspects of the Kalman Filter that can be changed to suit a different setup than the one described here.

3.1.1 Matrix Setup

When designing the A, B, C, and D matrices, the telescope system itself must first be accounted for since it can have major implications on how those matrices are constructed. The system used here, as explained in Section (3.1.3), handles the control effort. This system only requires inputs for right ascension and declination and will proceed to those coordinates itself. This affects the A and B matrices because the states for the telescope itself do not need to be factored into the equations of motion. Therefore, only the satellite’s dynamics need to be considered here. The A matrix used here is shown in Equation (3.1). The reason this A matrix will be used is because, as shown in Figure (3.1), the RA and DEC values for a satellite in a higher orbit will generally follow the equation of a line. Therefore, the velocity can be considered roughly constant for both RA and DEC. This is shown in Figures (3.1) and (3.2) which show the RA and DEC for 6 satellites that are in geostationary (GEO) orbit. The source of the data is explained in Section 3.4. The B matrix is a zero matrix because, since the control inputs are handled by the telescope itself, there is no need to factor any control effort into these state equations.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

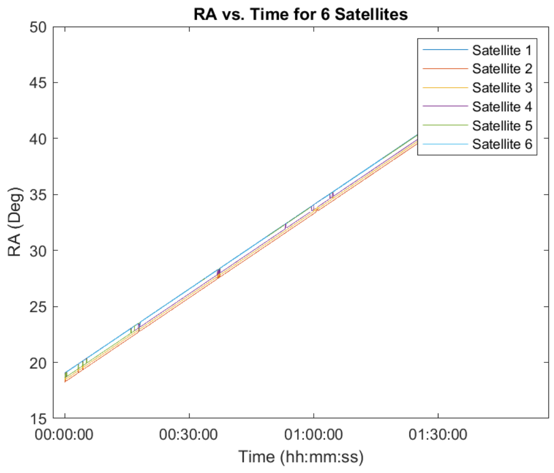

Figure 3.1 The RA over time for 6 GEO satellites.

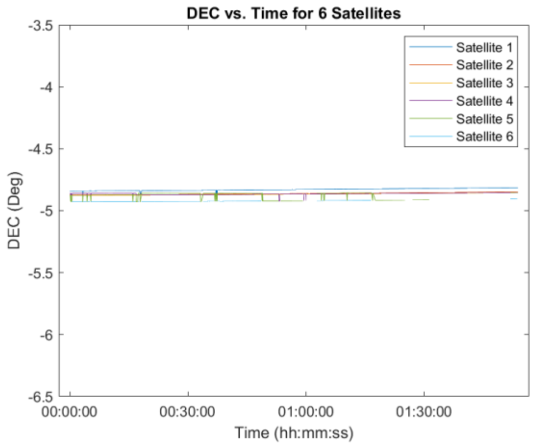

Figure 3.2 The DEC over time for 6 GEO satellites.

The C matrix is shown in Equation (3.2). The telescope system used here has the ability to measure the right ascension and declination angles as well as their velocities. However, only the angles are needed because the goal of the estimator is to predict the location of the satellite in the right ascension and declination coordinate system. The D matrix is also a zero matrix since there is no need to measure the control effort.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

To determine if the system is fully observable, Equation (2.12) was used to get the observability matrix shown in Equation (3.3). The rank for this matrix is 4 which means it is fully observable and the eigenvalues for the filtered system can be arbitrarily placed. In other words, it is at least theoretically possible to create a state estimator using a Kalman Filter.

$$P_o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

The next step is to determine the Kalman Gain. To use this method, $\hat{Q}$ and $\hat{R}$ must be determined. These are shown in Equations (3.4) and (3.5) as functions of the reported measurement noise, which is explained in more detail in Section 3.3. $\hat{R}$ is structured as a 2x2 matrix with the diagonal terms being functions of the measurement noise as reported by the telescope system's instruction manual. Although there is no information in the manual for process noise, a reasonable $\hat{Q}$ was created using the noise in this scenario. G is assumed to be a 4x4 identity matrix for mapping $\hat{Q}$ into the system. These layouts for G, $\hat{Q}$, and $\hat{R}$ were chosen because the results produced with the matrices showed sufficiently small error while showing an acceptable ability to correct itself after measurements were factored in after each time step.

$$\hat{Q} = \begin{bmatrix} \frac{noise}{10} & 0 & 0 & 0 \\ 0 & noise^2 & 0 & 0 \\ 0 & 0 & noise & 0 \\ 0 & 0 & 0 & noise^2 \end{bmatrix} \tag{3.4}$$

$$\hat{R} = \begin{bmatrix} noise^2 & 0 \\ 0 & noise^2 \end{bmatrix} \tag{3.5}$$

3.1.2 Image processing and Initial Satellite Recognition

For the image processing, there is only a need for 2 methods to filter noise out of the image. The Gaussian smoothing filter and the threshold filter, both described in Section 2.8.2, are used to filter the noise, and create a smoothed image. The image is put through these 2 filters before attempting to locate centroids and streaks to reduce the likelihood of noise getting labelled as a centroid or streak. The method utilized to find the centroids is the intensity-weighted method described in Section 2.8.3.

The initial method used in this research for locating a satellite in an image was to use the smoothed image, which would then be run through the Hough Transformation, described in Section 2.8.3, to get a set of candidate lines. These were then tested to ensure they did not cross within or near their endpoints. The first line is compared to the others by taking the equation of a line figuring out where it intersects with the others, one at a time. If these lines intersect between the endpoints of both lines, or reasonably close to the endpoints, then both lines are eliminated. The reason for this process is that, if the lines intersect close together, it can mean one of the lines, or potentially both, may be a result of noise or a star. After eliminating these lines, a line is chosen from the remaining lines for tracking over a short period. The criteria for selection can be altered based on certain parameters, the main 2 parameters being camera exposure time and the satellite's altitude relative to the observer. The first criteria is important because a shorter exposure time means the satellite will make a shorter streak and vice versa. The second criteria is important because an object orbiting at a higher altitude will typically make a shorter streak than at a lower altitude, all other factors being equal.

Once the line is selected, the endpoints are then moved using the corners that were detected using the Harris Corner Detection method described in Section 2.8.3. This is accomplished by finding the distance between one endpoint and all of the corners and finding the minimum value. If this value is below a set threshold, that corner becomes the new endpoint. This process is repeated until both endpoints have moved to a position such that there are no more corners that are below the threshold. To prevent the endpoints from potentially moving endlessly between corners, another criteria is added. This new criteria is that the endpoint at the “top” of the line, relative to the image, cannot move to a point that is below it. Likewise, the endpoint at the bottom of the line cannot move to a point above it. Once this process is finished, the new endpoints form a new line to be used for propagating the satellite forward for a short period of time. A comparison between this method and the final method is described in Section 4.1.

The final method for initially detecting a satellite starts similar to the process described above with one change. Instead of using the smoothed image in the Hough Transformation, a binarized version of the smoothed image is used instead. These results are shown in further detail in Section 4.1. Figure (3.3) shows an example of the output of using the Hough Transformation this way, with the image on the left showing the entire image and the image on the right being a close-up of the satellite showing that the lines are all concentrated on the streak of the satellite. Unlike the above method, however, a different criteria was used for deciding which lines would not be used. Instead of determining which lines crossed, the angle they make with the vertical axis was used. If the angle was above a set threshold, the line was removed from consideration. This eliminated lines formed when stars were close to the satellite streak. After this, the remaining lines would then be averaged to create a single line. This new line would then be propagated forward a small period in time to get the next position of the satellite.

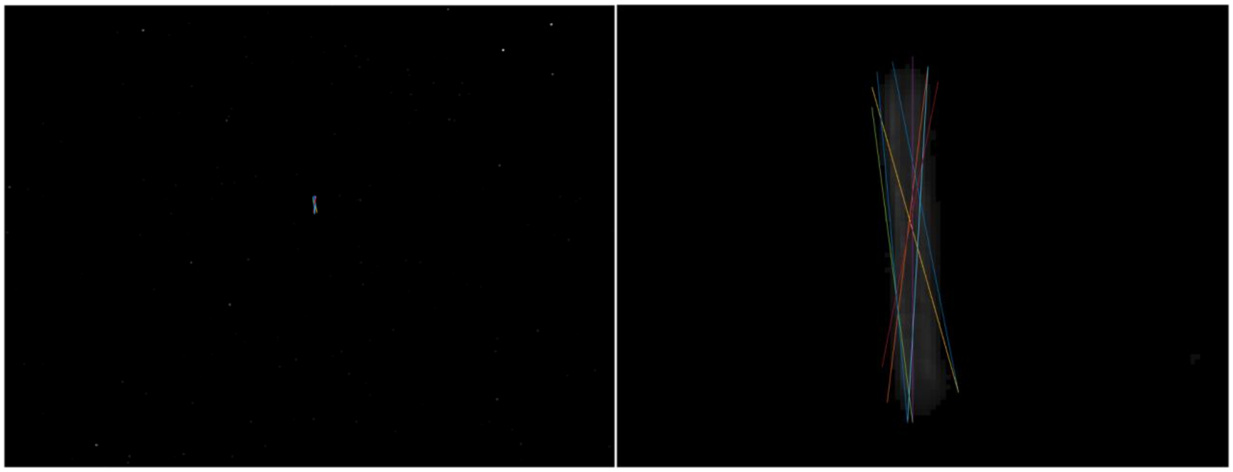

Figure 3.3 (Left) Image after the Hough Transform. (Right) Close-up of satellite.

The first step in propagating the satellite forward is to compute the change along the x-axis. This is done using Equation (3.6), where $\underline{x}_{avg}$ is the x-coordinates for the averaged line and $\underline{x}_{proj}$ is the projected x-coordinates. In the second term of the right hand side, dx is the difference in the x-coordinates of the averaged line and c is a constant that is determined by how far forward the satellite is propagated as a ratio of the change in time from the beginning of the first image to the beginning of the second image, or the end of the first to the end of the second, compared to the exposure time itself. If, for example, the exposure is 10 seconds and there is a gap of 10 seconds between images, the constant will be 2 since the satellite needs to be propagated 20 seconds into the future with an exposure time of 10 seconds. However, if the exposure time was 5 seconds and the gap between images was 15 seconds, the propagation time would still be 20 seconds, but the constant would instead be 4. For the first image, there is a small change to be made to how the constant is used. This change is to set the constant as $+c$ for one propagation and $-c$ for another propagation. It is necessary to do this twice since it is not yet necessarily known which direction the satellite is moving. For future images, this only needs to be done once in the direction the satellite is moving. Equation (3.7) shows the equation to propagate forward, where m is the slope of the averaged line and b is the y-intercept. Figure (3.4) shows this line and Figure (3.5) shows the projections.

$$\underline{x}_{proj} = \underline{x}_{avg} + c * dx \tag{3.6}$$

$$\underline{y}_{proj} = m * \underline{x}_{proj} + b \tag{3.7}$$

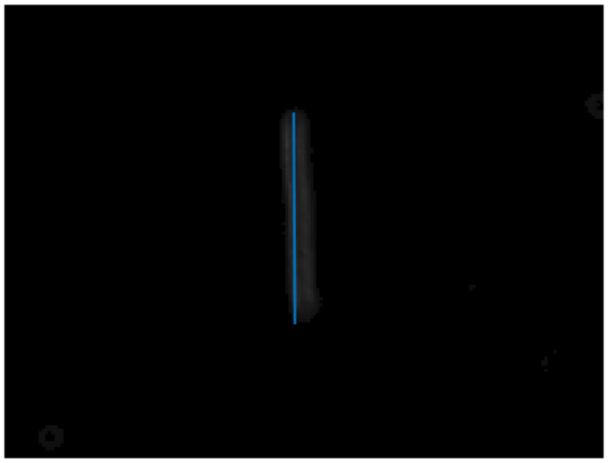

Figure 3.4 The resultant line after taking the mean of all Hough lines.

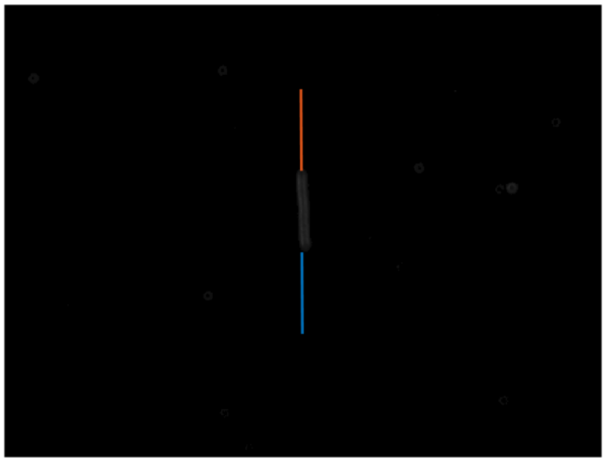

Figure 3.5 The propagation lines that result from the first image.

In subsequent images, the satellite needs to be found again. One method is to use the Hough Transformation again to get candidate lines. Also, same as before, lines are eliminated that do not meet a chosen criteria, which in this case will be if the angle made with the vertical axis exceeds a set threshold. Typically, if there is only one streak in the image, this will be enough to find the satellite again, though more criterion can be added as needed. In the case of one satellite in the second image, the next step is to find which projected line the streak is closest to.

One method is to find which projected line's endpoints are the closest to the endpoints of the line derived from the Hough Transformation. When multiple streaks lines in different spots of the image are present, a method that can be used is to find which of the Hough lines the projected endpoints are closest to and then only take the average of the lines in the immediate vicinity. If observing objects at a higher orbit such as GEO, only 2 images and the 4x4 system are needed. It is important to note here that the above methods for tracking the satellite across these images requires that the telescope and camera do not move. This is because the coordinate system used is relative to the image and will not translate between images if the camera moves. However, a transformation matrix such as the one in Equation (3.8) can be used to convert the endpoints from x-y to RA-DEC by simply having Astrometry determine the RA and DEC positions of any objects, such as the satellite and stars, and then taking any 2 of these centroids with their RA-DEC and x-y coordinates to satisfy Equations (3.9) and (3.10), which will give the constants needed to transform coordinate systems. Doing this on the first image allows the user to either convert back to x-y on the second image or to convert candidate endpoints to RA-DEC. The conversion back to x-y can be done by multiplying the left side of Equation (3.8) by the inverse of the 2x2 matrix (provided it is invertible) or by simply switching the 2 vectors so that the x-y coordinates are on the left side, remembering this will also switch the meaning of the variables.

$$\begin{bmatrix}\alpha\\ \delta\end{bmatrix} = \begin{bmatrix}\varphi & \Psi\\ \zeta & \Omega\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} \tag{3.8}$$

$$\alpha = \varphi x + \Psi y \tag{3.9}$$

$$\delta = \zeta x + \Omega y \tag{3.10}$$

3.1.3 The Kalman Filter

To generate the initial conditions, Aitken's and Neville's Algorithm, shown in Equation (2.56), is used to create a second order polynomial for both the ascension and declination angles, shown in Equations (3.11) and (3.12) respectively where α is the ascension angle and δ is the declination angle. These polynomials can be used to give the initial velocities and accelerations for the estimator by using the first and second derivatives. The initial velocities can be solved for by taking the derivatives of Equations (3.11) and (3.12). An example of these derivatives for the ascension is shown in Equation (3.13) where $\dot{\alpha}_0$ denotes the initial condition for velocity that will be used in the estimator. The initial angles will come from the second set of coordinates used in solving for the polynomials. This process is used twice, once for RA and once for DEC, to yield a total of 4 initial conditions.

$$a_1 \Delta t + a_0 = \alpha \tag{3.11}$$

$$b_1 \Delta t + b_0 = \delta \tag{3.12}$$

$$a_1 = \dot{\alpha}_0 \tag{3.13}$$

The next step is to propagate the states forward in time to get an estimate of where the satellite will be. The method used for estimation is the discrete-time Kalman Filter discussed in Section 2.6. This process starts with the last image used for acquiring the initial conditions and first involves assuming the initial updated covariance, P_k^+, which here is assumed to be a 4x4 diagonal matrix with each diagonal term being set to 0.001. At this stage, the updated state vector, $\underline{x}_k^+$ is set to the initial conditions. These are then propagated forward by a set Δt. This means first solving for the matrix F using Equation (2.39). As long as Δt remains constant, the same F may be used across all iterations. If Δt changes, then F must be solved for again.

The next step is to solve for $\underline{x}_{k+1}^{-}$ using Equation (2.42) and then to solve for P_{k+1}^{-} using Equation (2.43) For subsequent images, as soon as the measurement is ready, the process outlined in Figure (3.6) is used until the end of the test. This process starts by computing the Kalman Gain from Equation (2.44), followed by updating the state vector with Equation (2.45) and then computing the updated covariance with Equation (2.46). These are then used to propagate to the next iteration by solving for $\underline{x}_{k}^{-}$ and P_{k}^{-}.

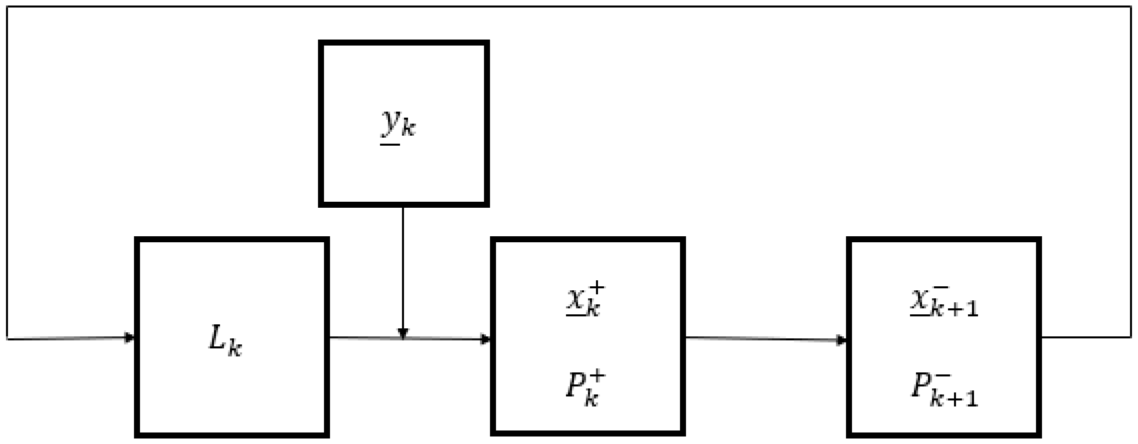

Figure 3.6 The discrete-time Kalman Filter.

3.2 Software

The primary software used in this research is Matrix Laboratory (MATLAB), made by MathWorks. All of the image processing, satellite recognition, and state estimation was done in this program with the exception of attaining RA and DEC values for images, which is explained below.

Astrometry is a software program that uses plate-solving as a method to determine what section of the sky an image shows based on star positions in the image compared to a database [16]. Astrometry is a Linux program, so it does require an interface program to be able to work with other operating systems. To interface with Windows, as was the case here, a Cyg-win must be used to get the Unix interface needed to be able to utilize Astrometry's functions. The particular astrometry configuration used here is provided by ANSVR [17].

For acquiring the images, a software program called TheSkyX Professional Edition was used. This is the software program that interfaces with the telescope mount. The software is capable to driving the telescope mount to coordinates set by a user as well as interfacing with the camera to control when it takes a picture and how long the exposure time is [18].

3.3 Hardware

The telescope mount that is used here is the Paramount MyT. As stated above, this telescope mount is capable of handling its own dynamics and the control effort. This means that only coordinates need to be input for the mount to operate. The mount was also used for determining a value for the variable $noise$ used in Equations (3.4) and (3.5). This value is assumed to be 4 arcseconds, $\frac{4}{3600}$ degrees of a circle. This assumption comes from the image system, where the scale is approximately 4 arcseconds per pixel [18].

The camera used for this research is the ASI1600MM Pro. This model is a monochrome camera, meaning its images will be in grayscale. This camera comes with a built-in cooling system capable of significantly reducing the temperature of the sensor depending on the current. For instance, at 0.5A, the camera sensor can be cooled to just below 32^{o}F from an ambient temperature of 84^{o}F. This means that there will be a reduction in the thermal noise in the image. The camera also has certain settings that are adjustable and help with improving the signal to noise ratio of images. It is recommended to set the camera gain lower for longer exposure pictures to have a higher dynamic range. For shorter exposure times, it is recommended to set this gain higher to reduce the read noise. Read noise includes circuit noise, analog to digital converter (ADC) quantization error noise, and pixel diode noise. Another adjustable parameter is the quantum efficiency (QE), which is the portion of the photons that are turned into electric signals in the camera [21]. This creates the image intensity. It is recommended that this is set to 60%. These two settings, when adjusted correctly for the images being collected, along with the cooling, account for much of what the frames mentioned in Section 2.8.1 attempt to account for. This is because the gain helps with electronic read noise, like the bias frame. The cooling system helps with the thermal noise, like the dark frame. Lastly, the QE along with the gain help to maximize the dynamic range of the camera for the image, like the light frame. This is the reason the frames were left out of the image processing portion of the process mentioned in Section 3.1 [20].

The telescope used here is a Rowe Ackermann Schmidt Astrograph (RASA) telescope with an 11-inch lens. This telescope was a good candidate for this research because it is designed for capturing images at night, especially for wide, flat-field imaging.[21]

3.4 Images and Testing Data

The images that were used for testing the Hough Transformation, as explained in Section 4.1, constitute images that were acquired for the purpose of testing the Kalman Filter in an end-to-end test and other images that were acquired for testing purposes. To get these images, the hardware setup from Section 3.3 is used and TheSkyX Professional Edition is used for software.

The data that is used in Section 4.2 for the purposes of testing a continuous-time system and for initial testing of the discrete-time system was acquired as part of research related to using a template matching method for tracking a satellite constellation. In this constellation, there are 6 satellites. However, the data for only 1 was used for testing the Kalman Filter across various values of Δt. It is important to note that, occasionally, the template matching system would incorrectly label some of the satellites as different satellites. This is useful for testing the Kalman Filter because it essentially provides a “disturbance” that could occur from an image processing technique labelling a star as the target satellite. Therefore, this provides an excellent opportunity to see how the Kalman Filter reacts to such a disturbance and whether it can regain the correct trajectory [22].

The images that are used for the end-to-end testing done in Section 4.3 were acquired using the hardware setup in Section 3.3 and TheSkyX Professional Edition software mentioned in Section 3.2. These images were captured across a 4 minute and 39 second timeframe. These images were captured in such a way that essentially mimicked the process in Section 3.1. This was done to simulate how the process would work in its entirety when implemented on a real system. Another advantage of acquiring images this way is that it allows for testing with real-world measurement and process noise. This is arguably more valuable than simulated noise since it allows the Kalman Filter to be run the way it is designed, which is for implementation.

4 Results

A variety of tests were conducted. The first test listed will pertain to image processing and the Hough Transform to show the results of changing the image that is used in the Hough Transform. The second set uses data that was acquired at a prior date to provide the ability to test the Kalman Filter in different scenarios to ensure it had the ability to follow the trajectory it is supposed to while adjusting itself to account for error, such as when another object is mistaken for the target satellite. The last set of tests utilized the full end-to-end process described in Section 3.1 to determine how it would handle the entire process of acquiring the data itself and then propagating.

4.1 Image Testing

The final method for recognizing satellites involved using a binarized version of an image after being smoothed with the Gaussian smoothing filter, known in this section as Process 2. The prior method involved using the smoothed image, known here as Process 1. Both were meant to be run through the Hough Transform, and Process 1 was also run through the Harris corner detection method to find corners in the image. These methods were tested on multiple images of varying exposures to see which would give the best chance at successfully discerning a satellite from the objects and noise in an image. The first set, shown in Figure 4.1, show the same image run through each method, with Process 1 on the left and Process 2 shown on the right. Both methods output 8 lines. However, Figure 4.2 shows a close-up of one of the lines from Process 1 between 3 stars, whereas Process 2 did not determine a line to be here. All 8 lines for Process 2 are concentrated on the satellite streak in the middle of the image. The 7 remaining lines for Process 1 are also concentrated on the streak.

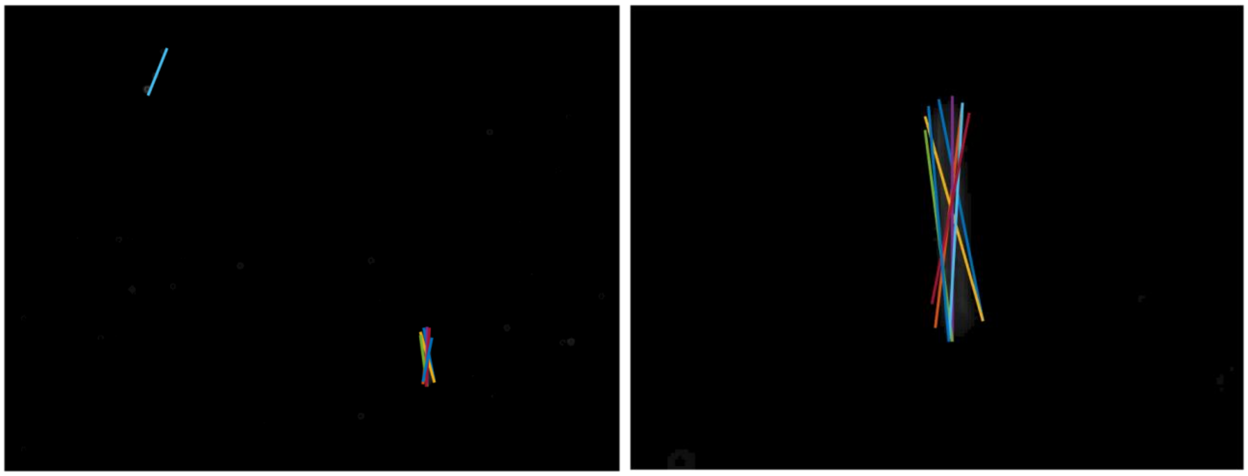

Figure 4.1 (Left) Result of Process 1. (Right) Result of Process 2.

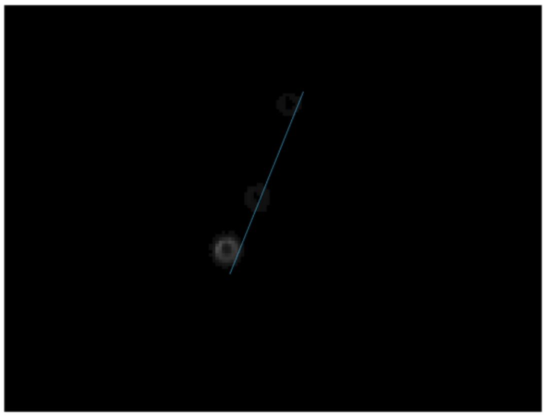

Figure 4.2 Close-up of one line from Process 1 that falls along 3 apparent stars.

Figure 4.3 shows the output of a particularly noisy image for each process. Process 1 determined that there were 194 lines, none of which were the satellite streak in the image. While Process 2 also did not identify the satellite streak with a line, it did only identify 105 lines. The importance of this is that it shows the Hough Transform is less sensitive to any remaining noise in an image when using the binarized image instead of the smoothed image. This decreases the chances of somehow factoring in a line that is not part of the satellite into any method used for recognizing the satellite.

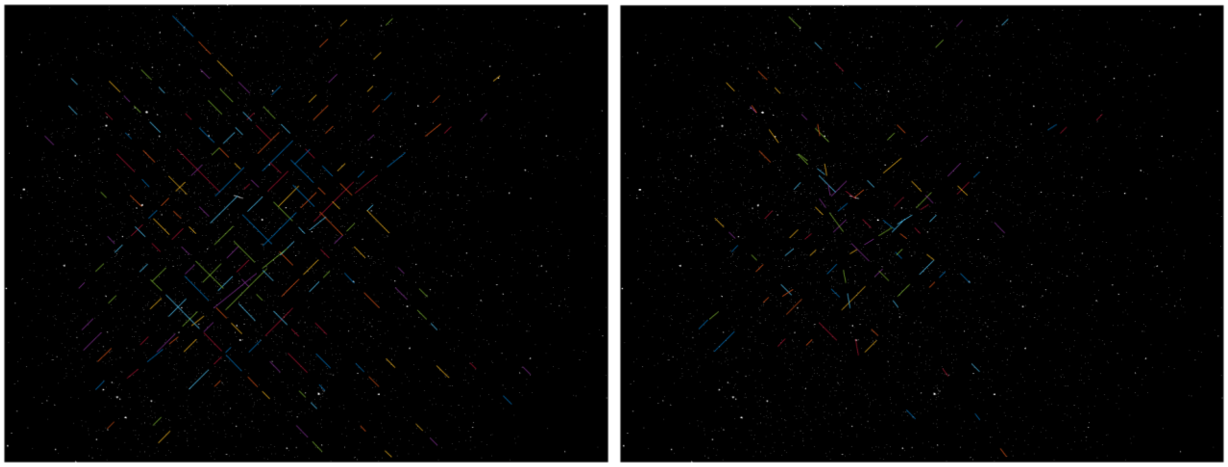

Figure 4.3 (Left) Process 1 on poorly filtered image. (Right) Process 2 on same image.

It is important to note that the reason these images retained as much noise as they did is because the threshold parameter for the noise filtering was set too low, meaning there was a significant amount of noise that made it through the image processing phase. When the thresholding parameter is raised from 2.5 to 4.5 for both processes, they both output 4 lines. The results are shown in Figure 4.4, where all 4 lines for Process 2 fall on the satellite streak whereas one of the lines for Process 1 falls on objects that can be labelled as either lines or noise.

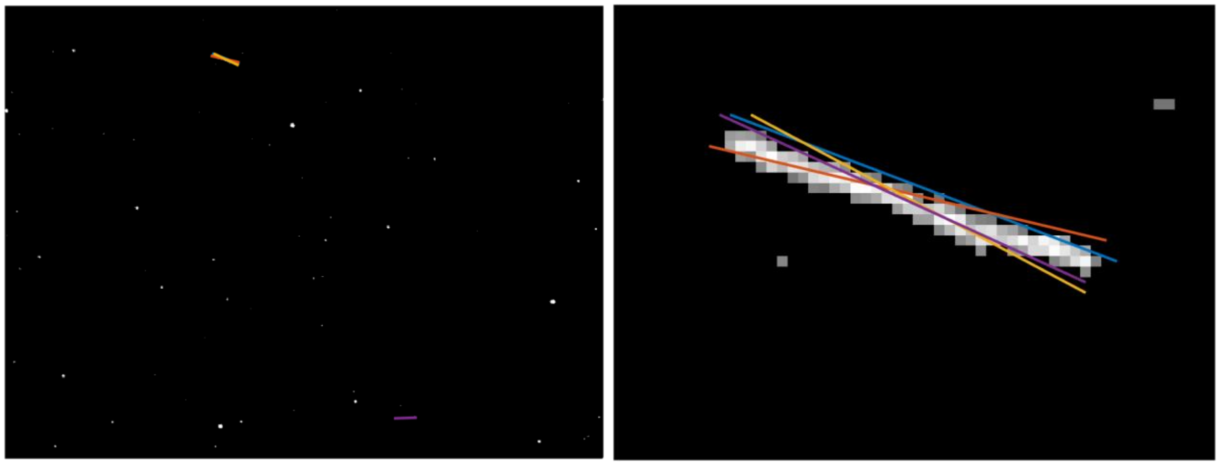

Figure 4.4 (Left) Lines from Process 1 after threshold increase. (Right) Lines from Process 2 for the same image.

4.2 Reliability Testing

To test whether it is possible to use a Kalman Filter in this scenario, a test was run using a continuous-time Kalman Filter that was designed using MATLAB's lqr function as described in Section 2.5. The resulting Kalman Gain is shown in Equation (4.1) and the corresponding eigenvalues for the estimator are shown in Equation (4.2). This method was tested using information on a satellite that was gathered prior, as explained in Section 3.4. Figures (4.5) and (4.6) shows the RA and DEC plots for this test.

$$L = \begin{bmatrix} 9.59 & 0 \\ 1 & 0 \\ 0 & 30.03 \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

$$\det(A - LC) = \begin{bmatrix} -0.03 \\ -0.11 \\ -9.49 \\ -30.00 \end{bmatrix} \tag{4.2}$$

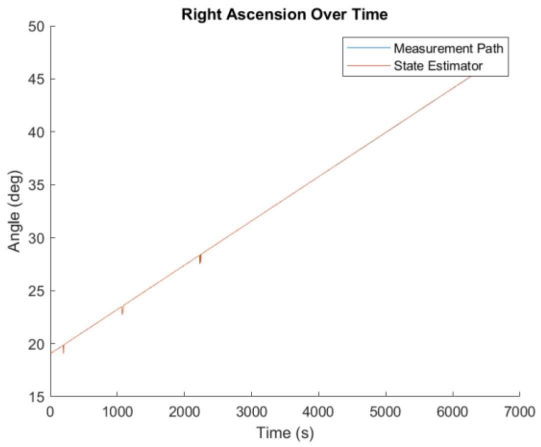

Figure 4.5 RA for the continuous-time test.

The error for this test, shown in Figure 4.7, shows, in more detail, that the continuous-time system was able to follow the measurements it was being fed since the error would change when the input was changed, but would return to 0. Although it may be difficult to see, this test shows that the filter was able to follow the path created by the measurements. Since this is a continuous-time test, the space between measurements was interpolated by MATLAB so that a continuous-time system could be used.

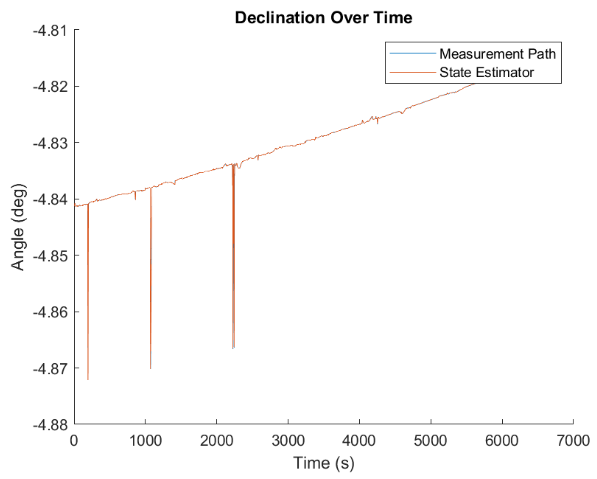

Figure 4.6 DEC for the continuous-time test.

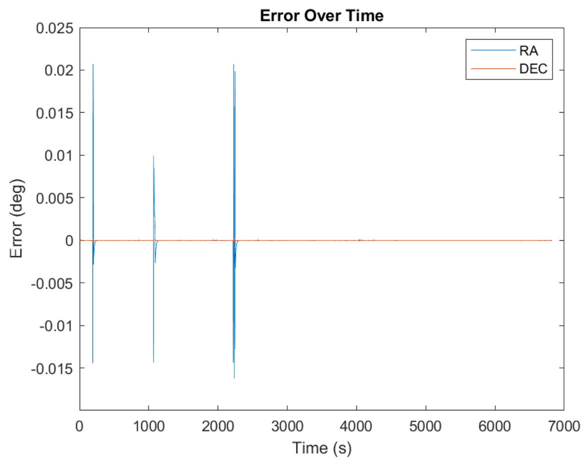

Figure 4.7 Error for the continuous-time test.

After the continuous-time system was tested, a discrete-time Kalman Filter was tested using the same satellite information as used in the first test. Initially, a test was run with a typical Δt of approximately 5.5 seconds to determine if the discrete-time Kalman Filter could keep the error small. However, the time between measurements would fluctuate occasionally. Figures 4.8 and 4.9 show the results of using this filter and Figure 4.10 shows the errors over time across the test run. The propagated and updated covariance plots are shown in Figure 4.11. The updated covariance plot shows that the Kalman Filter was able to successfully update the state vector to the measurements that were being given to it. The propagated covariance, on the other hand, has some spikes. This is expected since the spikes correspond to the times where measurement time increased significantly.

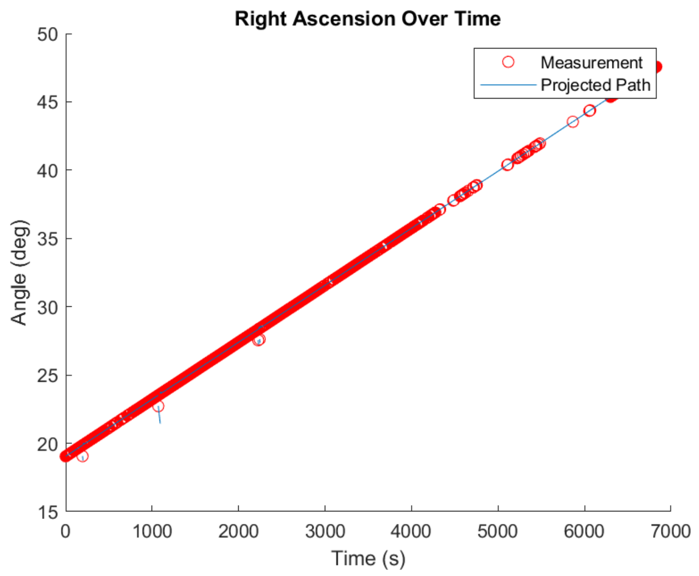

Figure 4.8 RA for typical Δt of 5.5 seconds.

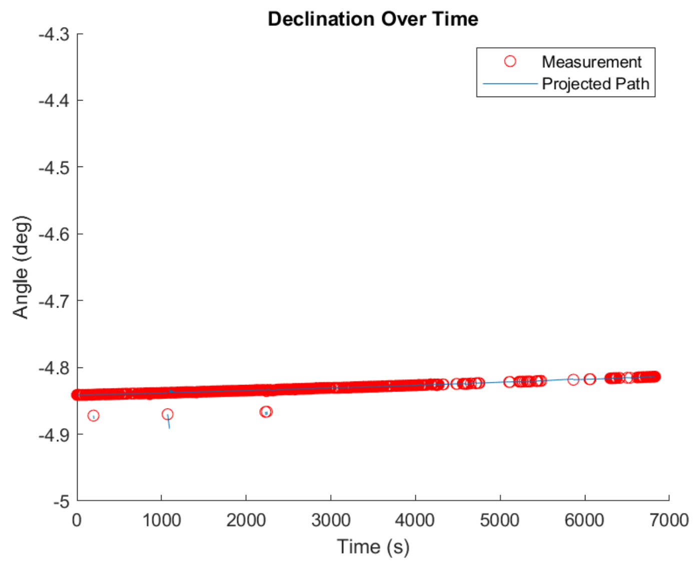

Figure 4.9 DEC for typical Δt of 5.5 seconds.

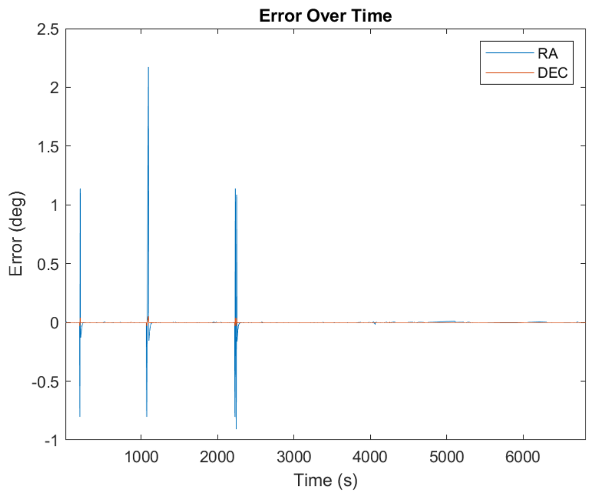

Figure 4.10 Propagation error for typical Δt of 5.5 seconds.

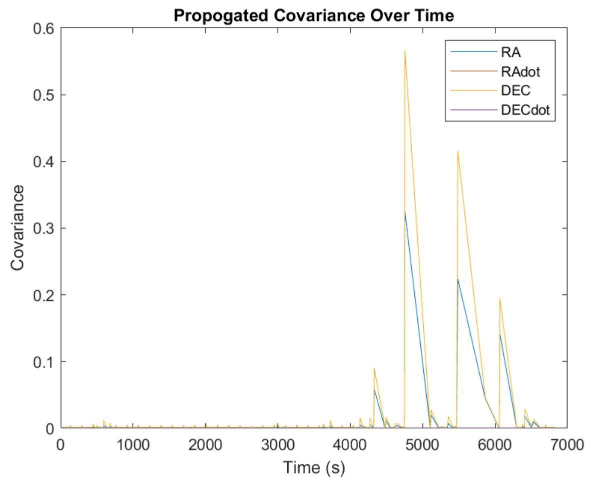

Figure 4.11 Covariance for propagation step for typical Δt of 5.5 seconds.

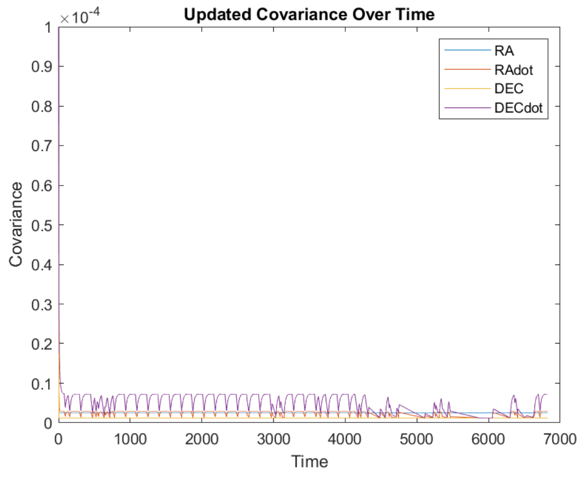

Figure 4.12 Covariance for update step for typical Δt of 5.5 seconds.

After testing with a small change in time between readings, another test was conducted with the same data, this time skipping over readings to only use every tenth measurement, for a typical measurement gap of approximately 1 minute. The results of this test are shown in Figures 4.13 and 4.14 with the error being shown in Figure 4.15. The covariance for the propagated step is in Figure 4.16 and the updated step shown in Figure 4.17. The updated covariance again shows the filter was able to correct the state vector based on measurements while the propagated covariance showed a similar behavior as the test before, with slightly higher magnitudes.

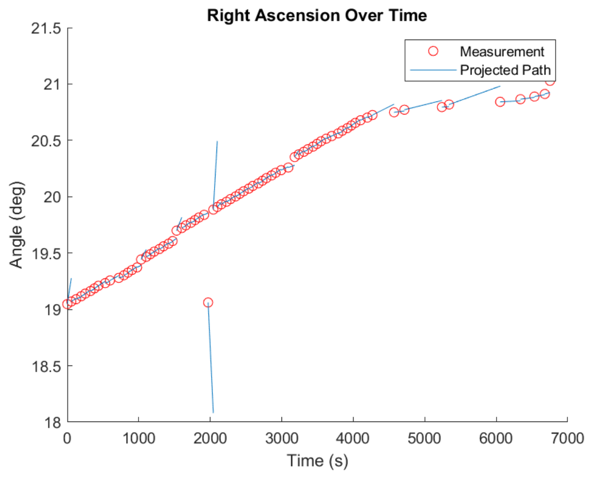

Figure 4.13 RA for a typical Δt of 1 minute.

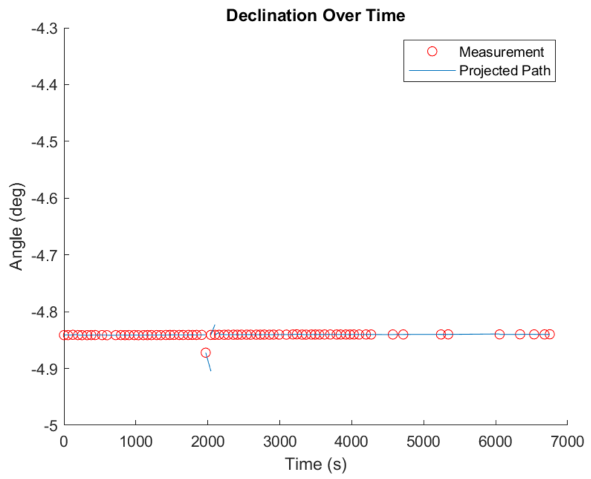

Figure 4.14 DEC for a typical Δt of 1 minute.

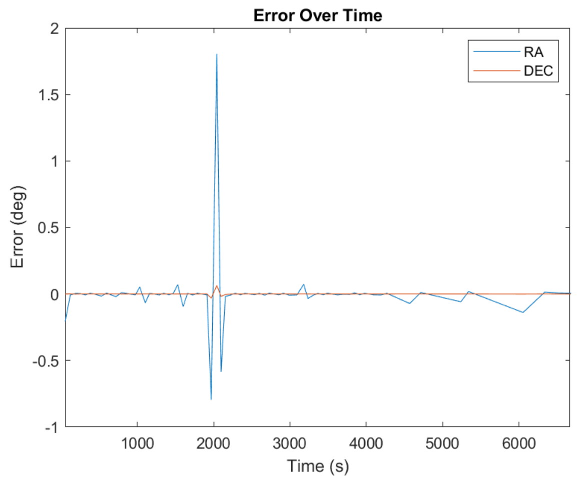

Figure 4.15 Propagation error for a typical Δt of 1 minute.

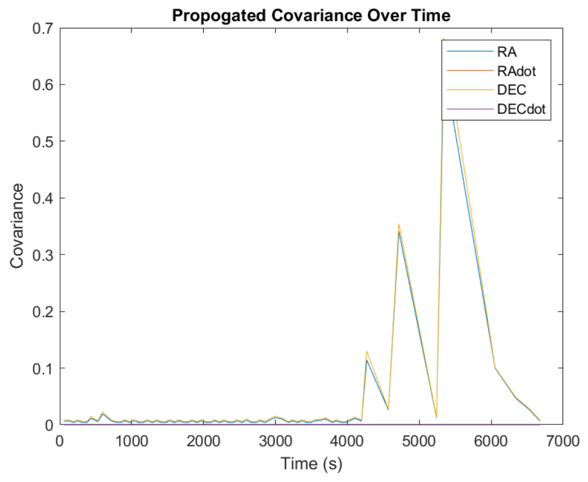

Figure 4.16 Covariance for propagation step for typical Δt of 1 minute.

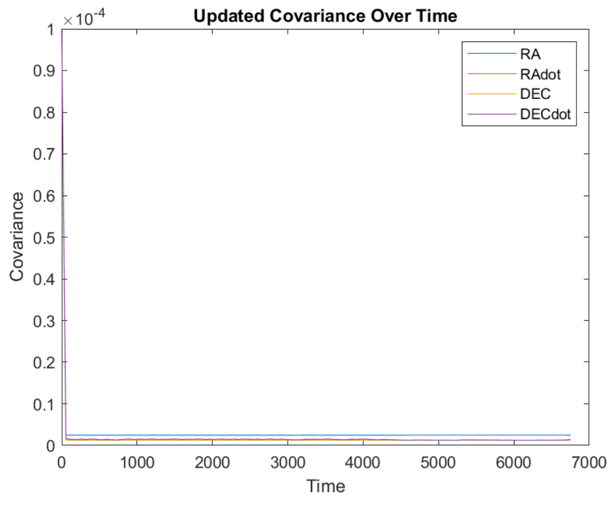

Figure 4.17 Covariance for update step for typical Δt of 1 minute.

4.3 Hypothesis Testing

After these tests were successfully performed, a full test was performed combining both the image processing and the Kalman Filter. The first run was performed on a set of 20 images that had a 10 second exposure time and were taken across a 4 minute and 39 second interval.

Figure 4.18 shows the measured and predicted RA values for this test on the left while Figure 4.19 shows the measured and predicted DEC values. These figures show that the Kalman Filter was able to successfully propagate the states forward and then update itself after each reading. Figure 4.20 shows the error between the projections and measurements.

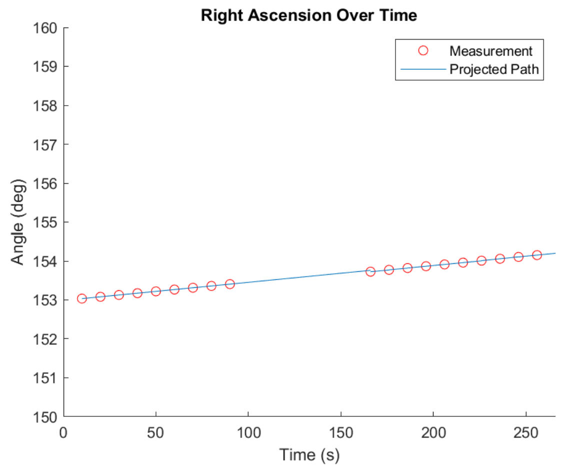

Figure 4.18 End-to-end RA for typical Δt of 10 seconds.

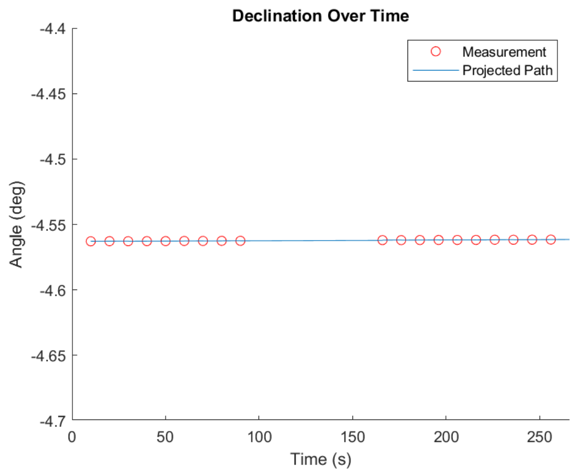

Figure 4.19 End-to-end DEC for typical Δt of 10 seconds.

The typical error for RA in the first 10 images is on the order of 10^{-4} with the exception of the very beginning, where the error is approximately -0.035 degrees. However, this error is smaller than the size of the field of view, which extends approximately 0.62 degrees from the center in the RA direction and 0.82 degrees from the center in the DEC direction. Figure 4.21 shows the propagated covariance, P_k^-, and Figure 4.22 the updated covariance, P_k^+. The updated covariance plot shows that the Kalman Filter was able to reliably update the current state after a measurement was included in the process, as the order of magnitude was 10^{-6}. Once again, the updated covariance plot shows that the filter successfully updated the state vector for the next propagation.

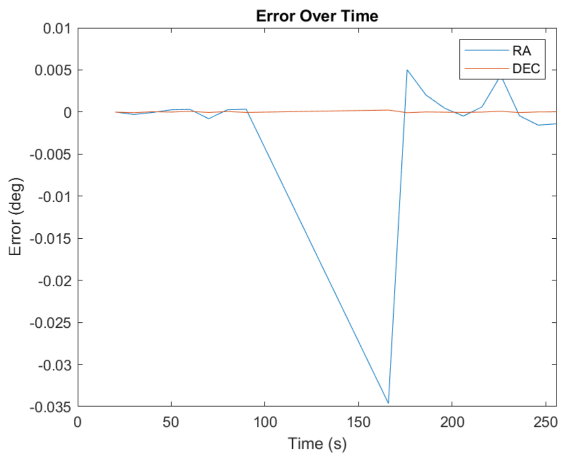

Figure 4.20 End-to-end error for typical Δt of 10 seconds.

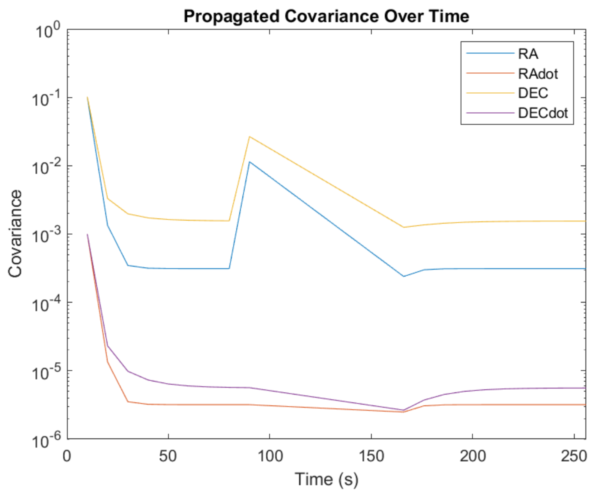

Figure 4.21 End-to-end covariance for the propagation step for typical Δt of 10 seconds.

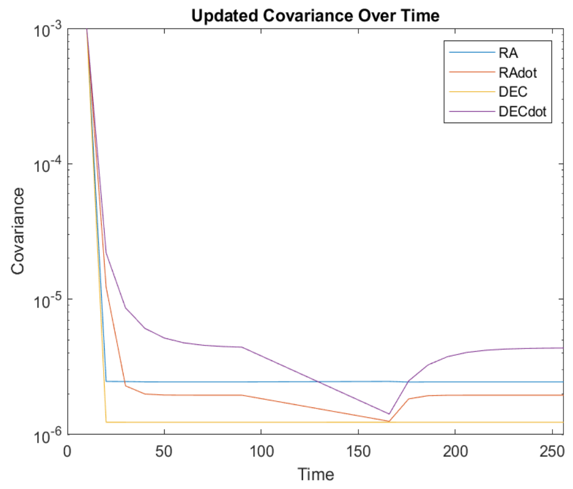

Figure 4.22 End-to-end covariance for the propagation step for typical Δt of 10 seconds.

Figures 4.23 and 4.24 show the results for the last hypothesis test for the Kalman Filter that was conducted. This used the same image set as above, except readings were skipped over. The initial satellite recognition used the same 2 first images as the above test and the same process. This time, the Kalman Filter was run with changing values for Δt. These values were, in seconds, 40, 40, 76, 40, and 50. The error is shown in Figure 4.25, where the highest magnitude for the error is approximately 0.15 degrees for RA, which falls after the first propagation. As stated above, the prediction needs to be within 0.62 degrees of the center of the image to at least be in the image for RA. Figure 4.26 show the propagated and updated covariances for this test. While the errors may have been larger than before, the updated covariance plot shows that the Kalman Filter was still able to correct the current state each time a measurement was made. The propagated covariance, however, rapidly settled with no spikes. This appears to show that, when looking at Figure 4.26 in combination with the earlier propagated covariance plots, there appears to be a trend where the spikes in the propagated covariance are more likely to occur with a sharp increase in Δt, with a larger increase in the magnitude correlating to a larger increase in Δt.

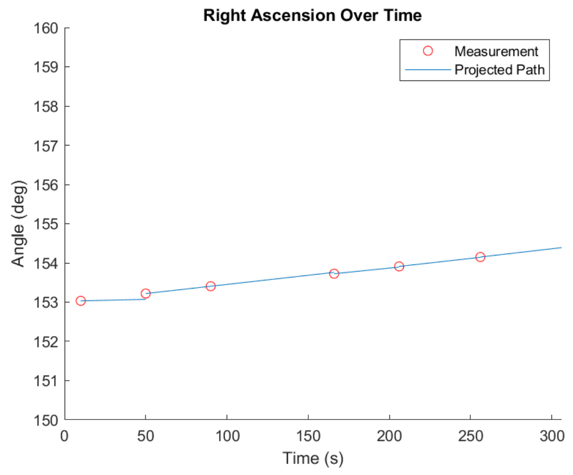

Figure 4.23 End-to-end RA for varying Δt of 40, 50, or 76 seconds.

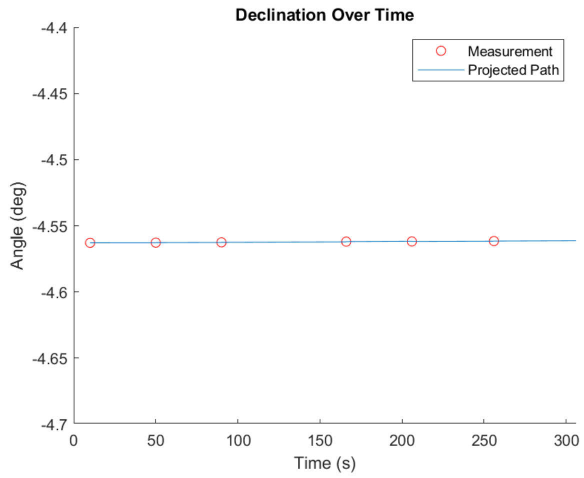

Figure 4.24 End-to-end DEC for varying Δt of 40, 50, or 76 seconds.

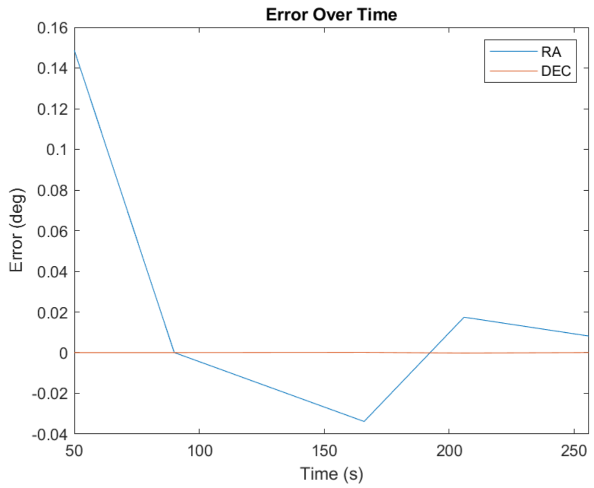

Figure 4.25 End-to-end error for varying Δt of 40, 50, or 76 seconds.

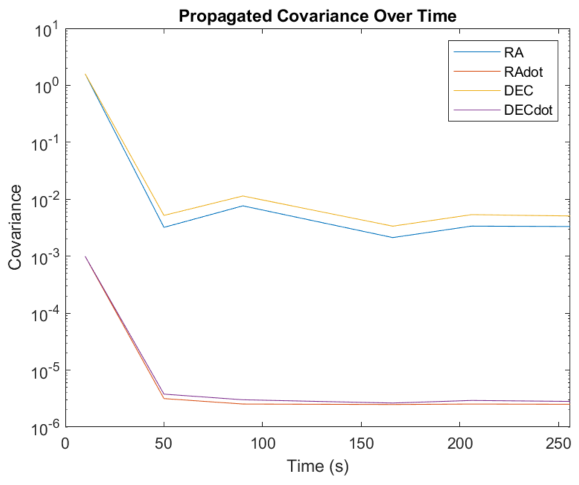

Figure 4.26 End-to-end covariance for propagation step for varying Δt of 40, 50, or 76 seconds.

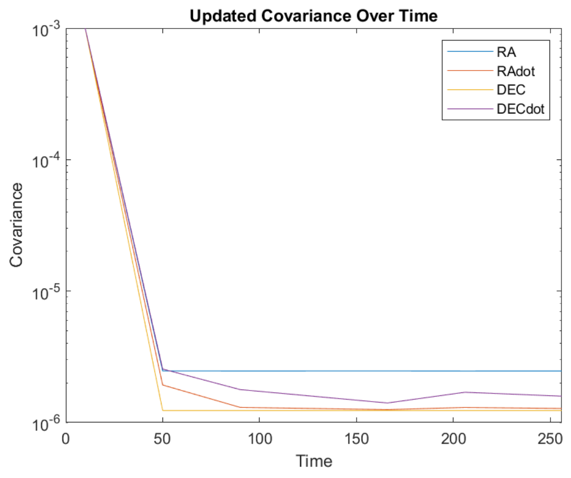

Figure 4.27 End-to-end covariance for the update step with varying Δt of 40, 50, or 76 seconds.

4.4 Image Testing

The last test run on the Kalman Filter was an implementation test using the software and hardware setup described in Sections 3.2 and 3.3. However, this test differs from the previous test because, instead of the image processing and satellite recognition technique described in Section 3.1.2, a neural network was used (see Ref [23]). The method used for the Kalman Filter, including acquiring the initial conditions, remained the same. Figures 4.28 and 4.29 show the results of this test.

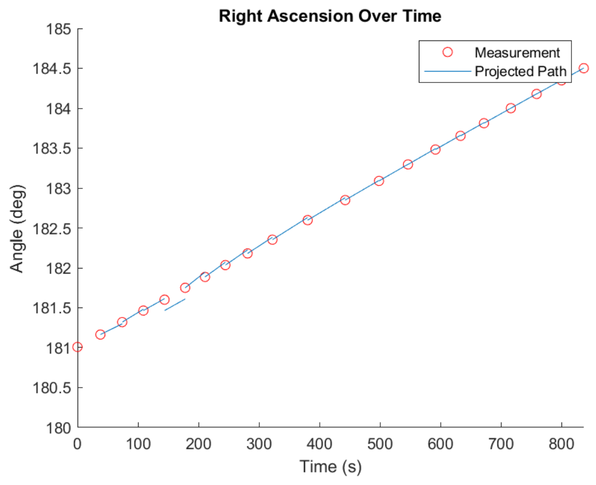

Figure 4.28 Right Ascension over time for the implementation test.

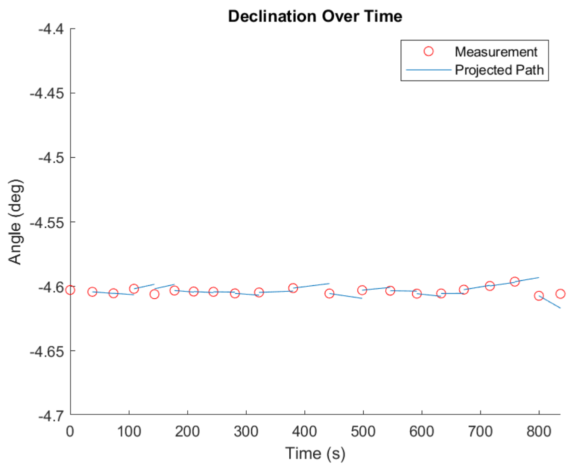

Figure 4.29 The declination over time for the implementation test.

Figure 4.30 shows the error across this test. Even though the magnitude of the error is consistently larger than earlier tests, the error did stay within the bounds set earlier of 0.62 degrees for RA and 0.82 degrees for DEC, with the largest error magnitude being less than 0.3 degrees for RA. The propagated covariance values stayed higher than the previous test for RA and DEC but stayed relatively close to the prior values for the updated covariance. This shows that the propagation has more expected deviation compared to prior tests, but the update step is still able to correct for the error.

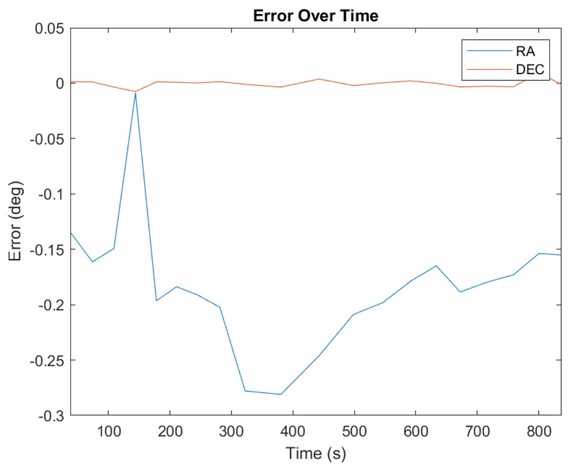

Figure 4.30 The error over time for the implementation test.

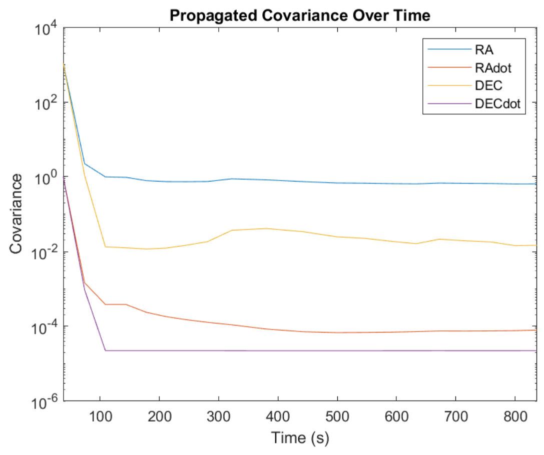

Figure 4.31 Propagated covariance for the implementation test.

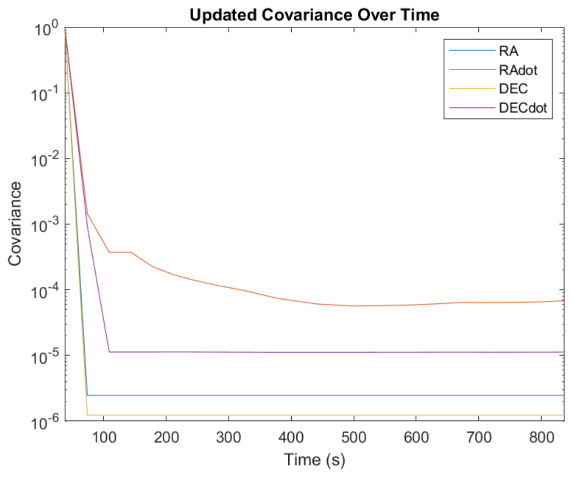

Figure 4.32 Updated covariance for the implementation test.

5 Discussions, Conclusions, and Recommendations

5.1 Discussion

The tests discussed in Section 4.1 demonstrated that switching the image that was used for the Hough Transform from the smoothed image to a binarized version of the smoothed image resulted in an improved ability to recognize the streaks that the satellites were making. The second test showed that if the threshold gain is set too low, namely for pictures with a short exposure time, that the binarized image resulted in fewer erroneous streaks connecting spots of noise and stars. In the other tests that were run, where the threshold gain was set sufficiently high for each particular image, the binarized image proved less likely to have excess streaks covering objects that were not the streak that the satellite made.

The tests done in Section 4.2 proved that the state estimator worked under a variety of conditions. The first test, using a continuous-time system, proved that using a Kalman Filter was possible under ideal circumstances. However, since continuous-time readings would not be possible to acquire under the current setup, a discrete-time system was devised and tested on the same satellite used for the continuous-time test, except the measurements were only accounted for as they had actually happened. Typically, this was approximately every 5.5 seconds, but some measurements had larger gaps. The Kalman Filter was able to successfully project the path that the satellite followed. The only times that it deviated from its course were times when the readings would be for a different satellite. However, despite the large error that would result from this, the filter would then correct itself once the readings were back to the correct satellite. The last test in this section involved skipping measurements so as to create larger values of Δt. The typical time between readings was approximately 55 seconds to 1 minute. This test showed that, even with larger values of Δt, the filter was still able to create reliable predictions as long as the measurements were for the correct satellite. Of note for the last 2 tests is that the updated covariance plots both show that the covariance remained low. This shows that the standard deviation is small since the standard deviation for each state is equal to the square root of the corresponding diagonal. This shows that the Kalman Filter was able to successfully update the current state to the measured values. That is a desired behavior for the filter since the measurements, with the setup utilized, are considered to be more reliable than the model.

Section 4.3 contains the end-to-end tests. The first test shows that the system was able to successfully propagate the satellite's position across the sky while keeping the error small enough to keep it well within the image. The determination of whether the Kalman Filter succeeded or failed comes from the error plot in Figure (4.25) and the updated covariance plot in Figure (4.27). The error plot shows that the predictions would always keep the satellite both within the image and close to the center. The updated covariance plot shows that, after a measurement is made, the system is able to correct itself so that the next propagation begins from the correct location.

Section 4.4 shows the final tests, when the Kalman Filter was implemented on a telescope to track a satellite in real time. The error plot for this test, Figure (4.30), shows that the filter was able to keep the satellite within the camera's field of view for the duration of the test. The updated covariance in Figure (4.32) then shows that, despite the error, the system was still able to update itself correctly. These results show that the filter successfully propagated the satellite's position and corrected any resultant error. This test also shows that the filter will work with other image processing techniques, provided the necessary coordinates can be obtained.

5.2 Future Work

Future work for this research is mainly comprised of 3 paths, changing the image processing and satellite recognition techniques used for other techniques and making the necessary changes and testing so that the estimator will work on lower orbiting bodies. Another possible path for future work would be to add to the entire process to account for the possibility of multiple satellites in the same image. Fortunately, the Kalman Filter is designed to be modular, with the full, end-to-end process having several methods, and parts of methods, that are interchangeable.

5.2.1 Image Processing and Satellite Recognition

As stated earlier, the Kalman Filter was designed so that it would only need the initial conditions and to be fed measurements after each image. Therefore, the image processing and satellite recognition methods can be changed. One possible change could include the implementation of SVD to reduce the noise in an image. This can be attempted along with the Gaussian filter that was used or as a replacement. Another possibility would be to perform corner detection or edge detection using SVD as a way to help generate projection lines.

For the satellite recognition portion, work can be done to help the system with handling multiple satellites in the same image. In its current form, it is designed to handle one satellite streak in an image. This can be done by either using a process to pick one satellite and follow that or to attempt to center the prediction on the middle of a cluster, if there is good reason to believe all the satellites are moving in the same direction at roughly the same rate. Something similar to the original method that was used for satellite recognition in Section 3.1, using the longer streak after eliminating any streaks that do not meet set criteria, is one possibility for picking out one streak out of multiple candidates.

5.2.2 Changes to the Dynamics

Implementing a 6-state system would involve first making changes to the A and C matrices as shown in Equations (5.1) and (5.2). The main limitation of the 4-state system is that it can only be used for higher orbits, such as GEO. Lower orbits, such as those classified as LEO, will form a parabolic arc when plotting their RA and DEC locations. Therefore, an extra state can be added for both RA and DEC to account for the constant acceleration needed to get this arc. Since the A and C matrices are changed in order to account for these extra states, the observability matrix must be recalculated using Equation (2.12). This result is shown in Equation (5.3) and has a rank of 6. Since the rank is equal to the number of states, this system is still observable.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.2}$$

$$P_o = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

Since the size of the A matrix has changed, $\hat{Q}$ will need to be changed as well. One possible example is shown in Equation (5.4). This matrix is essentially the same as Equation (3.4), but accounts for the process noise of the acceleration. Even though C has changed in size as well, $\hat{R}$ does not need to be changed since there are still only 2 measurements being recorded.

$$\hat{Q} = \begin{bmatrix} \frac{noise}{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & noise^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & noise^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & noise & 0 & 0 \\ 0 & 0 & 0 & 0 & noise^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & noise^2 \end{bmatrix} \tag{5.4}$$

Since this system has 2 extra states, there is a need for 2 extra initial conditions. Aitken and Neville's Algorithm can be used here to get all of the initial conditions. However, a third RA-DEC reading is required to be able to get enough data to use this algorithm. Equations (5.5) to (5.6) show the second order polynomials needed for RA and DEC. Equations (5.7) and (5.8) show the derivatives needed to solve for velocity and acceleration. The constants can be solved for using Equations (2.56) and (2.57).

$$a_2\Delta t^2 + a_1\Delta t + a_0 = \alpha \tag{5.5}$$

$$b_2\Delta t^2 + b_1\Delta t + b_0 = \delta \tag{5.6}$$

$$2a_2\Delta t + a_1 = \dot{\alpha} \tag{5.7}$$

$$2a_2 = \ddot{\alpha} \tag{5.7}$$

Since objects in lower orbits are in the sky for a shorter period of time, it is imperative to begin using the estimator as early as possible. In some cases, getting 3 separate images to be able to begin using the estimator may be impractical. However, as discussed in Section (3.1), it is possible to switch between the x-y coordinates of the image to RA-DEC. Therefore, by converting the endpoints of the satellite to RA-DEC and using the centroid of the streak as a third point, it is possible to obtain all of the needed initial conditions to begin using the estimator. A longer streak will produce more reliable initial conditions for the velocities and accelerations.

5.3 Conclusions

The use of a Kalman Filter to estimate the position of a satellite as it travels through the sky can be considered successful since it proved capable of handling every test that was implemented. In addition, the filter also proved capable of working with different image processing techniques. Lastly, because the Kalman Filter works as a method for tracking objects across the sky, by way of estimating their position and correcting error, this will give amateur astronomers, or maybe someone conducting their own AIOD research, another tool to draw from for their studies. It is worth noting, however, that the filter is only going to be as reliable as the measurements, which are a product of the techniques used for image processing and satellite recognition. Fortunately for those that wish to use this method for their own endeavors, they will easily be able to utilize newer, faster, algorithms as they present themselves as they create a version of what is presented here that best suits their own needs and the equipment they utilize.

6 REFERENCES

[1] Zuehlke, D., “Space Image Processing and Orbit Estimation Using Small Aperture Optical Systems”, Masters Thesis, Aerospace Engineering Department, Embry-Riddle Aeronautical University, Daytona Beach, FL, 2019, Embry-Riddle Aeronautical University Scholarly Commons, pp 1-6.

[2] “Orbit Outlook”, DARPA, (n.d.), Retrieved March 17, 2023. Retrieved from: https://www.darpa.mil/program/orbitoutlook

[3] Brogan, W. L., “State Variables and the State Space Description of Dynamic Systems”, *Modern Control Theory*, 2nd ed., Prentice Hall, NJ, 1985, pp 225-244

[4] Brogan, W. L., “Controllability and Observability for Linear Systems”, *Modern Control Theory*, 2nd ed., Prentice Hall, NJ, 1985, pp 306-308.

[5] Brogan, W. L., “Design of Linear Feedback Control Systems”, *Modern Control Theory*, 2nd ed., Prentice Hall, NJ, 1985, pp 397-417.

[6] Ogata, K., “Control Systems Analysis in State Space”, *Modern Control Engineering*, 5th ed., Prentice Hall, NJ, 2010, pp 687-805.

[7] Crassidis, J. L., Junkins, J. L., “Review of Dynamic Systems”, *Optimal Estimation of Dynamic Systems*, 2nd ed., CRC Press, NY, pp 597-599.

[8] Crassidis, J. L., Junkins, J. L., “Sequential State Estimation”, *Optimal Estimation of Dynamic Systems*, 2nd ed., CRC Press, NY, pp 143-149.

[9] “What You Need to Know”, *The Paramount Robotic Telescope System User Guide*, Software Bisque, Inc., December, 2016, pp 17-19. Retrieved from: https://buytelescopes.com/software-bisque-paramount-myt-gem-224018

[10] Zuehlke, D., “Space Image Processing and Orbit Estimation Using Small Aperture Optical Systems”, Masters Thesis, Aerospace Engineering Department, Embry-Riddle Aeronautical University, Daytona Beach, FL, 2019, Embry-Riddle Aeronautical University Scholarly Commons, pp 50-66.

[11] Harris, C., Stephens, M., “A Combined Corner and Edge Detector”, *Proceedings 4th Alvey Vision Conference*, 1988, pp 147-152

[12] Muthumalai, R. K., Gopalsamy, U., “Note on Neville Method of Interpolation”, International Journal of Pure and Applied Mathematics, Vol. 116, No. 23, 2017, pp 99-102. Retrieved from: https://acadpubl.eu/jsi/2017-116-23-24/articles/23/13.pdf

[13] Workalemahu, T., “Singular Value Decomposition in Image Noise Filtering and Reconstruction”, Masters Thesis, Department of Mathematics and Statistics, Georgia State University, Atlanta, GA, 2008, ScholarWorks @ Georgia State University, pp 3-10.

[14] Tang, J., Wang, Y., Huang, C. Liu, L., Al-Nabhan, N., “Image Edge Detection Based on Singular Value Feature Vector and Gradient Operator”, *Mathematical Biosciences and Engineering*, Vol. 17, Issue 4, 2020, pp 3721-3735. Retrieved from: http://www.aimspress.com/article/doi/10.3934/mbe.2020209

[15] Datta, B. N., “QR Factorization, SVD, and Projections”, *Numerical Linear Algebra and Applications*, Society for Industrial and Applied Mathematics, PA, 2010, pp 222.

[16] “Project Summary”, Astrometry.net, (n.d.), Retrieved August 4, 2022. Retrieved from: https://astrometry.net/summary.html

[17] Galasso, A., “ansvr – local Astrometry.net plate solver for Windows”, (n.d.) Retrieved August 4, 2022. Retrieved from: https://adgsoftware.com/ansvr/

[18] “The Paramount and TheSkyX Professional Edition”, *The Paramount Robotic Telescope System User Guide*, Software Bisque, Inc., December, 2016, pp 120-175. Retrieved from: https://buytelescopes.com/software-bisque-paramount-myt-gem-224018

[19] Dark Current, Teledyne Photometrics, Retrieved March, 17, 2023. Retrieved from: https://www.photometrics.com/learn/advanced-imaging/dark-current

[20] ZWO ASI1600 Manual, Suzhou ZWO Co., LTD., March, 2018. Retrieved from: https://astronomy-imaging-camera.com/product/asi1600mm-cool

[21] Berry, R., Celestron Engineering Team, Big! Fast! Wide! Sharp! The Story of the Rowe-Ackermann Schmidt Astrograph, Celestron, March, 2020. Retrieved from: https://celestron-site-support-files.s3.amazonaws.com/support_files/RASA_White_Paper_2020_Web.pdf

[22] Zuehlke, D., Tiwari, M., Henderson, T., “Autonomous Template Generation and Matching for Satellite Constellation Tracking”, *AIAA SCITECH 2022 Forum*, 2022 Retrieved from: https://www.researchgate.net/publication/357565859_Autonomous_Template_Generation_and_Matching_for_Satellite_Constellation_Tracking

[23] Jordan, J., Posada, D., Zuehlke, D., Radulovic, A., Malik, A., Henderson, T., “Satellite Detection in Unresolved Space Imagery for Space Domain Awareness Using Neural Networks”, *American Astronomical Society*, (Not Yet Published) Retrieved From: https://www.researchgate.net/publication/362252394_Satellite_Detection_in_Unresolved_Space_Imagery_for_Space_Domain_Awareness_Using_Neural_Networks