System Design and Implementation of the Virginia Tech Optical Satellite Tracking Telescope

Daniel Patrick Luciani

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Jonathan T. Black, Chair
Mazen H. Farhood
Craig A. Woolsey

26 April, 2016
Blacksburg, Virginia

Keywords: Optical Satellite Tracking, Earth-Satellite Viewing Geometry, Satellite Surveillance, Statistical Orbit Determination

System Design and Implementation of the Virginia Tech Optical Satellite Tracking Telescope

Daniel Patrick Luciani

ACADEMIC ABSTRACT

The Virginia Tech Optical Satellite Tracking Telescope (VTOST) aims to test the feasibility of a commercial off-the-shelf (COTS) designed tracking system for Space Situational Awareness (SSA) data contribution. A novel approach is considered, combining two COTS systems, a high-powered telescope, built for astronomy purposes, and a larger field of view (FOV) camera. Using only publicly available two-line element sets (TLEs), orbital propagation accuracy degrades quickly with time from epoch and is often not accurate enough to task a high-powered, small FOV telescope. Under this experimental approach, the larger FOV camera is used to acquire and track the resident space object (RSO) and provide a real-time pointing update to allow the high-powered telescope to track the RSO and provide possible resolved imagery. VTOST is designed as a remotely taskable sensor, based on current network architecture, capable of serving as a platform for further SSA studies, including unresolved and resolved imagery analysis, network tasking, and orbit determination. Initial design considerations are based on the latest Raven class and other COTS based telescope research, including research by the Air Force Research Lab (AFRL), ExoAnalytic Solutions, and other university level telescope projects. A holistic system design, including astronomy, image processing, and tracking methods, in a low-budget environment is considered. Method comparisons and results of the system design process are presented.

System Design and Implementation of the Virginia Tech Optical Satellite Tracking Telescope

Daniel Patrick Luciani

PUBLIC ABSTRACT

The Virginia Tech Optical Satellite Tracking Telescope (VTOST) is designed to meet a number of multi-faceted objectives. First, VTOST is being used to test the ability of a commercial off-the-shelf (COTS) based tracking system to gather quality satellite data, and contribute to networks that maintain knowledge of Earth orbiting objects. This would allow these over tasked networks to offload data collection to sites such as VTOST. A novel approach, on the university level, is considered, combining two COTS systems, a high-powered telescope, built for astronomy purposes, and a larger field of view (FOV) camera. Under this experimental approach, the larger FOV telescope is used to acquire and track the satellite using novel image processing techniques, and provide a real-time pointing update to allow the high-powered telescope to track the object and provide additional information.

In addition to providing object tracking support and validating novel image processing techniques, VTOST is a remotely taskable system which acts as a platform to test tasking algorithms. Finally, VTOST is a multi-purpose tracking testbed that can be used by Virginia Tech space researchers that would benefit from real-world data and implementation, including image analysis, collecting real-world satellite data for orbit determination (OD), testing new optical detection and tracking methods, and other future studies.

Acknowledgments

I would like to thank Dr. Black for all of his advice and guidance during my graduate studies. Also, I would like to thank Dr. Farhood and Dr. Woolsey for taking time to be on my committee and for helping me develop my dynamics and control knowledge and interest during my undergraduate and graduate studies.

Additionally, I would like to thank Dr. J.P. Laine and the rest of his group at C. S. Draper Laboratory for the incredible learning experience during my internships there and for introducing me to statistical OD problems – I am grateful to them for allowing me to find my passion.

I was lucky to have another incredible learning experience and introduction to the SSA field through my internship at the Air Force Research Laboratory (AFRL) at AMOS. I would like to thank Dr. Chris Sabol and Dr. Paul Schumacher for their time answering my many questions and Dr. Kim Luu for the opportunity and her guidance.

I would like to thank my lab mates for allowing me to continuously bounce ideas off of them, my siblings for support, and my friends for remaining so throughout this work.

Lastly, I would like to thank my parents for all of their continuing support – I am lucky and forever grateful.

Contents

List of Figures

List of Tables

List of Abbreviations

AFRL Air Force Research Laboratory.

AMOS Air Force Maui Optical and Supercomputing Site.

ASCOM Astronomy Common Object Model.

CCD Charge Coupled Device.

CIO Celestial Intermediate Origin.

CMOS Complementary Metal-Oxide Semiconductor.

COTS Commercial Off-the-Shelf.

DARPA Defense Advanced Research Project Agency.

ECEF Earth-Centered Earth-Fixed Reference Frame.

ECI Earth Centered Inertial Reference Frame.

EKF Extended Kalman Filter.

EOP Earth Orientation Parameters.

ESpOC Exoanalytic Space Operations Center.

FOV Field of View.

GCRF Geocentric Celestial Reference Frame.

GEM German Equatorial Mount.

GEODETICA General Electro-Optical Detection, Tracking, Identification, and Characterization Tool.

IAU International Astronomical Union.

ICRF International Celestial Reference Frame.

ICRS International Celestial Reference System.

IERS International Earth Rotation Service.

ISS International Space Station.

ITRF International Terrestrial Reference Frame.

JFCC Space Joint Functional Component Command for Space.

JSpOC Joint Space Operations Center.

LEO Low Earth Orbit.

LST Local Sidereal Time.

NGA National Geospacial-Intelligence Agency.

NORAD North American Aerospace Defense Command.

OD Orbit Determination.

ODE Ordinary Differential Equation.

RSO Resident Space Object.

SEZ South, East, Zenith Reference Frame.

SGP4 Simplified General Perturbations 4 Model.

SSA Space Situational Awareness.

SSN Space Surveillance Network.

TCRF Topocentric Celestial Reference Frame.

TEME True Equator Mean Equinox Reference Frame.

TLE Two-Line Element Set.

USNO U.S. Naval Observatory.

VTOST Virginia Tech Optical Satellite Tracking Telescope.

WGS-84 World Geodetic Reference System 1984.

Chapter 1

Introduction

1.1 Motivation and Previous Work

The Virginia Tech Optical Satellite Tracking Telescope (VTOST) is designed to meet a number of multi-faceted objectives. First, VTOST will be used to test the feasibility of contributing collected data to networks that maintain Space Situational Awareness (SSA). One such network is the Defense Advanced Research Projects Agency's (DARPA) Orbit Outlook, a program that seeks to improve SSA by adding additional data from more diverse sources [1]. Another is the Exoanalytic Space Operations Center (ESpOC) which aims to show the feasibility of non-dedicated sensor networking contributions to SSA [2]. SSA describes the field of observing, characterizing, and analyzing resident space objects (RSOs)- the accuracy of tracking and cataloging is a main priority [3]. The U.S. National Space Policy calls for effective global SSA [4], and recent space events, including the 2007 Chinese anti-satellite test [5] and the 2009 Iridium-Cosmos satellite collision [6], have shown the need for increased SSA capabilities.

U.S. Strategic Command's Joint Functional Component Command for Space (JFCC Space), through its Joint Space Operations Center (JSpOC) tracks approximately 20,000 RSOs that are greater than 10 cm in diameter, including active and inactive satellites, orbital debris, and spent rocket bodies, using approximately 400,000 observations each day [7]. This research will test the feasibility of a commercial off-the-shelf (COTS) designed telescope for SSA data contribution. If proven effective, low cost, COTS telescopes could be used to track bight, low earth orbit (LEO) RSOs, allowing the higher-powered sensors that make up the Space Surveillance Network (SSN) to track more difficult to observe RSOs.

In addition to providing SSA support, VTOST will be a remotely taskable sensor, based on current network architecture, which will act as a platform to test optimal tracking, network tasking, and orbit determination algorithms. Finally, VTOST will be a multi-purpose tracking testbed that can be used for resolved imagery analysis such as observing on-orbit fuel burns for maneuver detection, collecting real-world RSO telemetry for orbit determination (OD), testing new optical detection and tracking methods, and other studies, undertaken by Virginia Tech space researchers that would benefit from real-world data. The main requirement of VTOST is that it must be built with existing Virginia Tech equipment and facilities and COTS hardware. Additionally, VTOST operational code must be easy to understand, implement, and maintain. The software systems will be based on proven algorithms, new algorithms will be derived where necessary.

Recent research into COTS-based telescopes, for SSA purposes, started with the Raven-class telescope development at the Air Force Research Laboratory (AFRL) Directed Energy's Air Force Maui Optical and Supercomputing (AMOS) site. Research on Raven-class telescopes at AMOS advanced from proof of concept to a contributing sensor to the SSN [8]. Raven-class telescopes are a design paradigm in which COTS hardware and state-of-the-art image processing are combined for specific applications to complete defined mission requirements

[9]. Ravens-class telescope have been designed for an increasing number of applications, including attitude determination [10], debris observation [11], and high eccentricity orbit determination [12]. These and other previous COTS telescope designs are considered in the new VTOST design. Finally, research on quantifying accuracy and implementing standardization of reduction techniques for external, or not normally contributing, sensor data is currently underway, including work by Bellows [13].

VTOST seeks to advance this Raven-class line of research by developing a taskable sensor with the requirements listed above. A novel tracking approach is considered and implemented: combining two COTS telescopes, a high-powered telescope, built for astronomy purposes, and a larger field of view (FOV) telescope. Using only publicly available two-line element sets (TLEs), orbital propagation accuracy degrades quickly with time from epoch and is often not accurate enough to task a high-powered, small FOV telescope. Under this experimental approach, the larger FOV telescope is used to acquire and track the resident space object (RSO) and provide a real-time pointing update to allow the high-powered telescope to track the RSO and provide possible resolved imagery.

1.2 Overview

The purpose of this thesis is twofold: 1) to present the results of the COTS based telescope system design and novel tracking research and 2) to summarize the background information, algorithms, and operational procedures needed for VTOST tasking and operation. As much of the underlying mathematics in this thesis is geometry based, a brief summary of the reference frames used is presented. This thesis will present all pertinent design considerations: due to the scope of topics involved, including astronomy, astrometry, image processing, optical systems, instrumentation, and astrodynamics, relevant literature review is presented

within each design consideration section.

The second chapter presents hardware considerations and decisions including FOV and resolved imagery analyses. The third chapter presents viewing geometry considerations, including reference frames and ground-track and illumination studies. The third chapter is followed by the surveillance chapter which includes a treatment of orbit propagation and reference frame transformations, and presents surveillance implementation results. Next image processing research, dynamic modeling, and current hardware limitations are presented in the tracking chapter. A closed-loop tracking solution is derived for the current hardware setup. Real-time statistical OD is then presented, including background Extended Kalman Filter (EKF) research, measurement model and covariance transformation derivations, and filter simulation results. Conclusions and future work are then presented and, finally, installation and operational procedures and MATLAB function listings are presented in the Appendices.

Chapter 2

Hardware

2.1 Wide FOV Camera

2.1.1 Camera Specifications

The camera used in the wide FOV setup is The Imaging Source's DMK 41BU02 Monochrome Camera. The considerations on choosing this camera included satellite size in pixels, cost, size, proprietary software for compatibility with MATLAB, and type and producer of the image sensor. Bruski et. al designed a low cost system with similar considerations [14]. A charge coupled device (CCD) versus complementary metal-oxide semiconductor (CMOS) image sensor analysis, for low cost applications, was discussed with members of the Sensors and Imaging Group at C. S. Draper Laboratory [15]. This camera is about a 0.05 m cube, connects to the control computer through a universal serial bus (USB) 2.0 connection, and has a C lens mount. Table 2.1 lists camera specifications. See The Imaging Source's datasheet for additional camera information [16].

Table 2.1: Wide FOV Camera Specifications

Resolution	1280 x 960 pixels
Dynamic Range	8 bit
Sensitivity	0.05 lux
Exposure	1e-4 to 30 seconds
Gain	0 to 36 dB
Offset	0 to 511
Saturation	0 to 200%
White Balance	-2 to 6 dB
Temperature (Operation)	-5 to 45 °C
Temperature (Storage)	-20 to 60 °C

This camera's image sensor is the Sony ICX205AL diagonal 8 mm progressive scan CCD [17]. The image plane has a resolution of 1280 by 960 pixels and each pixel is a square with a length of 4.65 μm.

The lens used for this camera is the Nikon AF-S DX NIKKOR Lens. This lens has a 35 mm focal length, maximum aperture of F1.8, and is compatible with an F lens mount. Since the camera has a C mount, a Nikon F to C mount lens adapter is used. This lens was chosen due to its large maximum aperture when compared to other lenses in the same price range. The limiting observable apparent magnitude for this setup has been determined experimentally to be approximately magnitude 7 vmag. The visual magnitude scale is a logarithmic, normalized measure of brightness. The units expression, vmag, will be used for clarity.

The camera is then fastened onto an ADM Accessories Losmandy "D" Series Saddle using a procedure described in Appendix A. Figure 2.1 shows the camera, lens, and mount, wide FOV system.

Figure 2.1: Wide FOV System

2.1.2 FOV Analysis

After review of the CCD, lens adapter, and lens, an initial FOV approximation can be calculated, ignoring Spherical geometry for a small FOV, in a manner analogous to the method presented by Wertz, in [18]. Figure 2.2 shows the FOV geometry. The equations are evaluated for initial considerations with the lens, nominal $f = 35$ mm.

$$2\psi_1 = 2\tan^{-1}\left(\frac{2.98}{f}\right) = 9.7^\circ$$

$$2\psi_2 = 2\tan^{-1}\left(\frac{2.23}{f}\right) = 7.3^\circ$$

$$\text{FOV} = 9.7^\circ \times 7.3^\circ \tag{2.1}$$

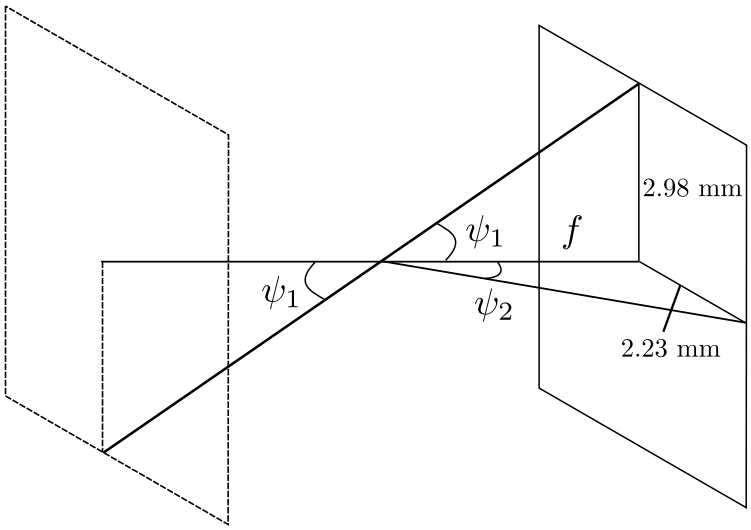

Figure 2.2: CCD and Field of View Geometry

Once an approximate FOV is calculated, the FOV in kilometers and meters per pixel, mapped to a specific orbit height, can be calculated. For an orbit height of 400 km, starting with the arc-length formula where $r = 400$ km and ψ is in radians:

$$S_1 = r2\psi_1 = 67.9 \text{ km}$$

$$S_2 = r2\psi_2 = 50.9 \text{ km}$$

$$\text{FOV} = 67.9 \times 50.9 \text{ km} = 1280 \times 960 \text{ pixels}$$

$$\frac{67.9 \text{ km}}{1280 \text{ pixels}} = 53.0 \frac{\text{meters}}{\text{pixel}} \tag{2.2}$$

Thus, for example, the International Space Station (ISS), which is about 109 x 73 meters on its largest side [19], would ideally be 2 x 2 pixels in an image. Modeling the real-world effects of atmospheric light scattering, non-perfect focusing, and lens imperfections on this measurement is beyond the scope of this project. A discussion was held with members of the Sensors and Imaging Group at C. S. Draper Laboratory [15], who stated that the real-

world effects on the ideal LEO meters per pixel for this camera, calculated above, would be acceptable for viewing the brighter RSO targets of this project. Experimental data has confirmed this conjecture, the ISS measures approximately 7 x 7 pixels in images taken with this camera setup.

Using an astrometric reduction using real images from the camera, star separation distances are used to accurately determine the camera FOV. The formulation above can then be reversed to find a highly accurate estimate of the focal length, which is needed for subsequent derivations. All astrometric reductions in this report are performed using Astrometry.net software [20]. Further information on Astrometry.net will be presented in section 5.1.1. The Astrometry.net solution for the image presented in Figure 2.3 is a FOV of $9.78875° \times 7.34581°$. Backing out the focal length from equation 2.1 gives an effective focal length of 34.75 mm.

Figure 2.3: Calibration image for Astrometry.net FOV calculation

Table 2.2: Camera Default Parameters

Brightness	0
Exposure	2 (4 sec)
Gain	1000
Gamma	20

2.1.3 Control Software

The camera is connected to the control computer through a USB 2.0 A Male to B Male connection cable. Once the camera is connected and the drivers are installed, see Appendix A, the camera is controlled via Microsoft ActiveX control by creating a camera ActiveX object in MATLAB. The camera object initialization function is presented in Appendix B. On initialization certain camera parameters that remain constant for all passes are set. These parameters were determined through calibration while observing the sky over Blacksburg. Units match those used by the camera MATLAB ActiveX object. These parameters are presented in Table 2.2

2.2 Celestron Telescope

2.2.1 Telescope Specifications

The base mount and telescope used in VTOST is the Celestron CGE Pro 1400 HD Computerized Telescope. This telescope is managed by Dr. John Simonetti in the Virginia Tech Physics Department for the Virginia Tech Astronomy Club, who has granted his permission for this research. Telescope operators must be certified by the Virginia Tech Astronomy Club and follow telescope operational procedures, presented in Appendix C. Table 2.3 lists perti-

Table 2.3: Celestron Telescope Parameters

Diameter (Aperture)	356 mm
Focal Length	3910 mm
Max Slew Speed	5°/second
Tasking	Az/El or α/δ
FOV	dependent on eyepiece/camera connection
Magnification	dependent on eyepiece/camera connection

nent telescope specifications. Additional telescope information can be found in the Celestron Datasheets [21] and [22].

The telescope has a German Equatorial Mount (GEM), which for this application is important because this means that one telescope rotation axis is nearly aligned with Earth's rotation axis. This geometry is investigated further in section 5.3.

2.2.2 Control Software

The Celestron mount is connected to the control computer through a RS-232 to USB adapter. The RS-232 adapter cord is connected to the bottom of the mount hand controller. Once the mount is connected and the drivers are installed, see Appendix A, the mount is controlled via Microsoft ActiveX control by creating a mount ActiveX object in MATLAB. The mount object initialization function is presented in Appendix B. Once the mount is initialized, it can be tasked with pointing commands.

2.2.3 Telescope Image Analysis

Initial magnification and FOV analysis, for possible resolved imagery, were performed by coupling the telescope with a borrowed Cannon EOS T2i (550D) DSLR Camera. The camera

is connected to the telescope using a Celestron T-Adapter and a, camera make specific, T-Ring connector. This camera has a image resolution of 5184 x 3456 pixels and a pixel size of 4.31 μm.

This setup produces a magnification of approximately 78, estimated by dividing the focal length of the telescope, 3910 mm, by the length of the T-Adapter, about 50 mm. This magnification results in an approximate FOV of 0.33° x 0.22°, which is found by dividing the apparent field of view of the camera by the magnification. This FOV results in a value of 0.444 meters per pixel, mapped to an orbit height of 400 km.

Preliminary object image plane size, telescope focus, and exposure times were considered by observing Jupiter. Jupiter and the ISS have a similar brightness, with a maximum magnitude of approximately -3 vmag [23]. Additionally, they are a similar apparent size in the sky. Image simulation was investigated to estimate the size of the ISS in the coupled telescope and camera FOV. Campbell, in [24], developed a FOV simulator for defined imaging components. Campbell's simulator was verified by comparing simulation images of Jupiter to images taken with VTOST. Figure 2.4 shows the simulated image of Jupiter and the collected image of Jupiter. The collected image of Jupiter was taken with an ISO 400 and an exposure time of 1/320 seconds. If this exposure time causes blurring in collected RSO images, the exposure time could be reduced and a stack of images could be created to achieve the brightness needed for image processing.

Thus, with similar brightness to Jupiter, the short exposure time would allow the ISS to be imaged during tracking with limited blurring. However, as Figure 2.5 shows, the ISS is small in the FOV. To minimize the meters per pixel, with low-cost hardware, the Celestron NexImage 5 Solar System Imager Camera and a Celestron X-Cel LX 1.25" 3x Barlow Lens is recommended for VTOST. With this proposed setup, which has an image resolution of 2592 x 1944 pixels and a pixel size of 2.2 μm, the FOV is estimated as 0.03° x 0.02°. Mapping

this FOV to an orbit height of 400 km results in 0.08 meters per pixel. Figure 2.5 shows a simulation of the ISS for the current telescope and camera setup, and for the proposed camera. Note that the number of pixels in the top image is roughly double, so the meters per pixel decrease with the proposed setup is not as pronounced as the figure shows. The meters per pixel with the proposed setup is approximately 5.7 times better. This analysis shows that once tracking is implemented, resolved imagery of the ISS is possible with the current hardware setup and proposed camera.

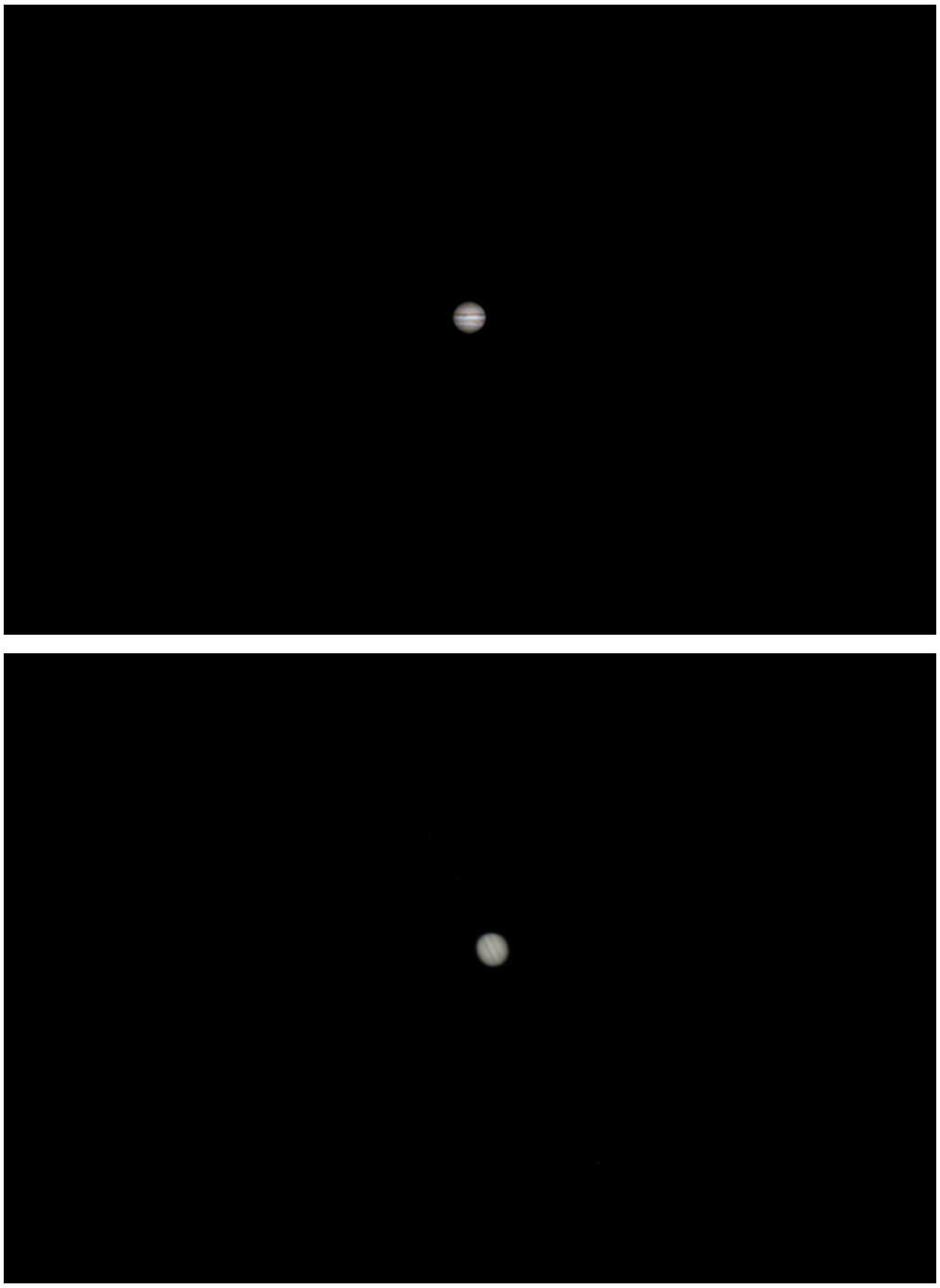

Figure 2.4: Top: Simulated Image of Jupiter, Bottom: Collected Image of Jupiter

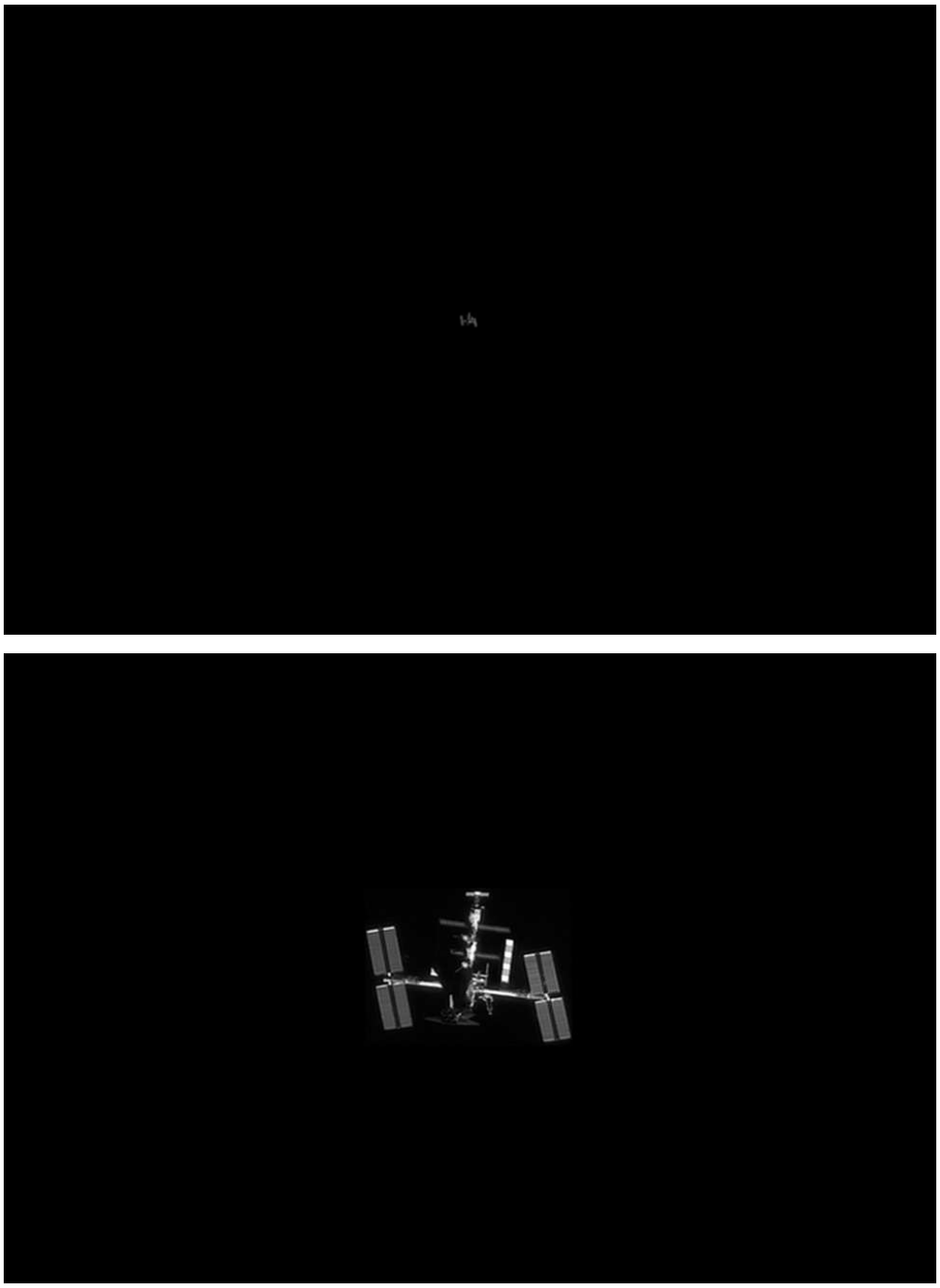

Figure 2.5: Top: Simulated Image of ISS, Bottom: Simulated Image of ISS, Proposed Setup

Table 2.4: Observatory Coordinates

Geodetic Latitude	37.2221°
Longitude	279.4599°
Height (msl)	644.7 m

2.3 Observatory

The telescope is located at the Nicholas R. Anderson Observatory off of Prices Fork Road, in Blacksburg, Virginia. The coordinates of the observatory, which are used in reference frame transformations in later chapters, are listed in Table 2.4. The height listed is height above mean sea level. The observatory has a motorized dome and dome slit that, currently, are only operable through levers. The dome is not currently coupled with telescope pointing, and thus, must be manually controlled for observation. To couple the dome and telescope pointing, the angular velocity of the dome and the local azimuth angle of the dome's home position must be measured. Then the time required for the dome to rotate from its current azimuth to any required pointing azimuth, solved for in section 4.2.3, can be calculated and commanded. Future steps for implementing fully autonomous operation include dome controller design.

Table 2.5 presents an inventory of VTOST hardware.

Table 2.5: Hardware Inventory

Component	Quantity
Imaging Source Camera	2
Nikon Lens and Filter	2
Lens to Camera Adapter	2
Camera to Telescope Connector	1
Camera to Connector, Nut and Screw Sets	2
RS-232 Connector	1
Celestron T-Adapter	1

Chapter 3

Viewing Geometry

Understanding specific viewing geometry is essential to predicting when a satellite can be viewed from a certain location on Earth, for example, in this application the Blacksburg Observatory. Low-cost optical sensors cannot image during daylight, when sunlight will dominate the exposure. Therefore, visibility is determined by two factors: the satellite must be within view of the observatory site, and the viewing location on Earth must be in twilight while the RSO of interest must be in sunlight. First, as much of the underlying mathematics in this thesis is geometry based, a brief summary of the reference frames used is presented. This chapter will present the background information and algorithms required to determine these two factors.

3.1 Reference Frames

3.1.1 Geocentric Celestial Reference Frame (GCRF)

The inertial frame used in this application is derived from the International Celestial Reference System (ICRS), a solar system barycentric system, defined by the International Earth Rotation Service (IERS) in [25]. Feissel, in [26], distinguishes a system as a "set of prescriptions and conventions together with the modeling requirements to define, at any time, a triad of axes", while a frame is a realization of a coordinate set within a system. The International Astronomical Union (IAU) has adopted the ICRS as the standard reference system. The International Celestial Reference Frame (ICRF) is a frame realized in the ICRS using extragalactic radio sources and is thus independent of solar system dynamics, an advantage over previous IAU standard frames [27]. Finally, the Geocentric Celestial Reference Frame (GCRF) was introduced by the IAU as the geocentric counterpart to the ICRF. Henceforth, the GCRF will be used as the inertial frame for this application. The GCRF can be visualized as the first axis aligned with vernal equinox and the third axis aligned with Earth's north pole (alignment is exact only for the date 1 January 2000). This reference frame is commonly referred to as Earth Centered Inertial (ECI), however, for real-world applications a specific inertial frame must be specified. For the Earth-satellite problem in this application, the GCRF is considered sufficiently inertial.

Right ascension, α, and declination, δ, are defined in the GCRF frame. Right ascension is the angle from $\hat{e}_1$ to the projection of the position vector onto the equatorial plane, measured positive to the east. Declination is the angle from the equatorial plane to the position vector, measured positive northward. Figure 3.1 shows the GCRF and an arbitrary position vector to show right ascension and declination.

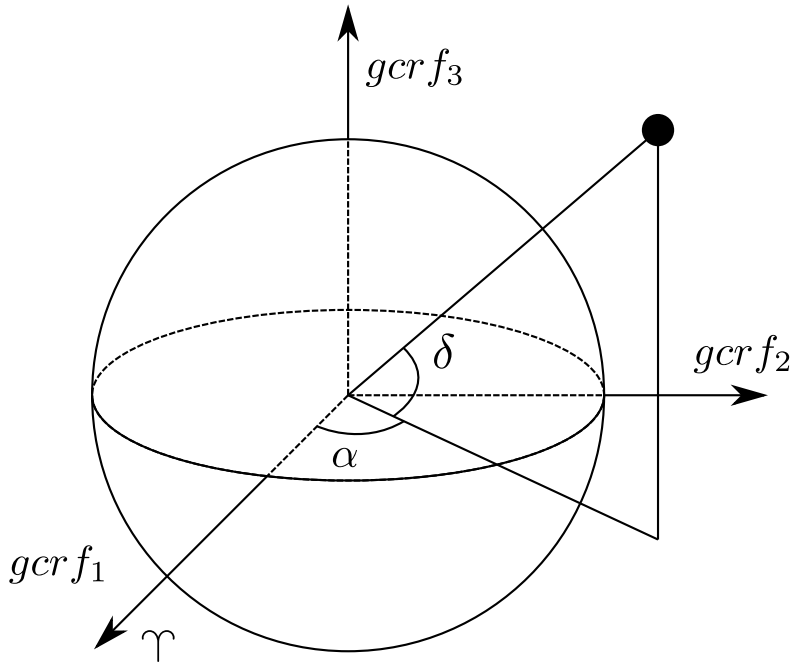

Figure 3.1: Geocentric Celestial Reference Frame

3.1.2 International Terrestrial Reference Frame (ITRF)

The International Terrestrial Reference Frame (ITRF), defined by the IERS in [25], will be the terrestrial frame used in this application. A terrestrial frame is fixed to the rotating earth, i.e., a location on Earth's surface is fixed in the ITRF, if plate tectonic motion is not considered. The U. S. Department of Defense primarily uses the World Geodetic Reference System 1984 (WGS-84) terrestrial frame which agrees with the ITRF within the uncertainty of the WGS-84 frame. The ITRF can be visualized as $\hat{e}_1$ aligned with the point of zero degrees latitude and longitude, and $\hat{e}_3$ aligned with Earth's north pole. This reference frame is often referred to as Earth Centered Earth Fixed (ECEF), however, ECEF is not specific enough for real-world applications.

3.1.3 Additional Reference Frames

A convenient extension of the ITRF is a rotation to an observation site's local south, east, and zenith directions and translation of the origin to the site location. This reference frame is known as the Topocentric Horizon Coordinate Frame and will be abbreviated as (SEZ). SEZ is a convenient frame when considering local measurements such as azimuth, which is defined as the angle measured from the north horizon point, positive to the east, and elevation.

Finally, a Topocentric Celestial Reference Frame (TCRF) is used. This frame is GCRF aligned with the origin translated from the center of the earth to a local site. This frame is encountered when considering satellite parallax and during astrometric reductions and when image processing is used to calculate a satellite position in the inertial frame with respect to the stars, as will be explained further in section 4.2.1. Topocentric right ascension and declination are defined in a manner analogous to geocentric right ascension and declination and are denoted with subscript t, α_t and δ_t, respectively.

3.2 Observable Satellite Sub-Latitude

First, an initial prediction for observable satellite sub-latitude points is derived for a circular orbit. A circular orbit is a reasonable assumption for most of the brighter LEO RSOs, especially as a first approximation. This technique solves for observable geocentric sub-latitudes, however, for the approximate observable small circle, geocentric and geodetic latitudes are assumed to be equal. Geodetic latitude geometry is presented in the next section. In this thesis, all latitude is assumed to be geodetic latitude unless otherwise specified.

Wertz, in [28], presents a derivation of satellite sub-latitude viewing geometry, based on the

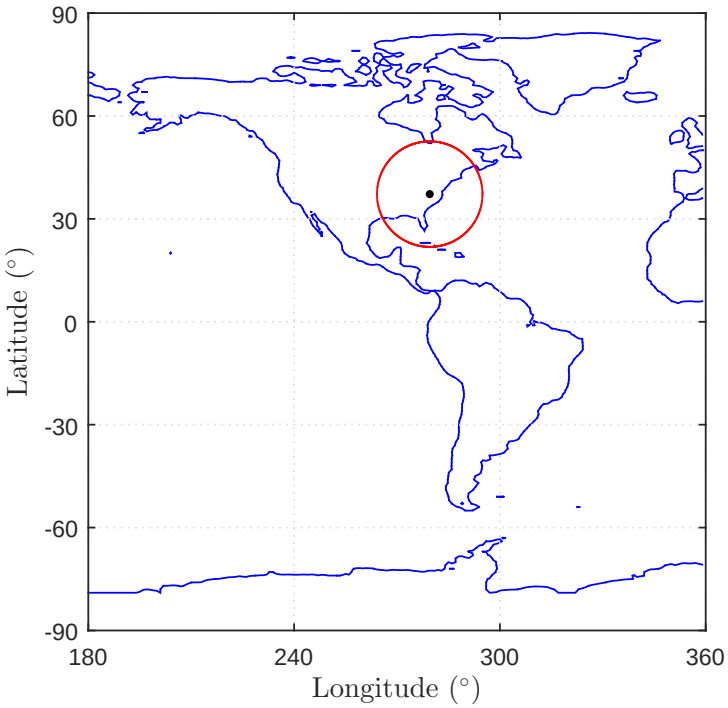

Figure 3.2: Observable Satellite Sub-Latitude Small Circle

assumptions that the ground trace is a great circle arc, i.e., a spherical earth, which can be summarized as follows:

$$sin(\eta_{max}) = \sin\left(\frac{R_\oplus}{R_\oplus + h}\right)\cos(\epsilon_{min})$$

$$\lambda_{max} = \tfrac{\pi}{2} - \epsilon_{min} - \eta_{max} \tag{3.1}$$

where h is the orbit height of the satellite, ϵ_{min} is the minimum local elevation for which viewing is possible, assumed to be 10 degrees for this application, and λ_{max} is the radius of the small circle centered on the Blacksburg Observatory, located in the figure as a black dot, that represents the set of observable satellite sub-latitude points. This viewing small circle, for an orbit height of 400 km, is shown in red in Figure 3.2.

3.3 Satellite Ground Track

Next, the satellite sub-latitude points must be determined as a function of time, commonly referred to as calculating the satellite ground track. The satellite position vector is found in GCRF through orbit propagation: for this application the Simplified General Perturbations 4 (SGP4) orbit propagator is used. SGP4 is presented in section 4.1. The position vector is then transformed from GCRF to ITRF by a process presented in section 4.2.2. Once the position vector in ITRF at every timestep is known, the geodetic sub-latitude point can be calculated. Care must be taken when considering earth oblateness effects: if a spherical rather than ellipsoidal earth model is used, the error along the longitude arc can be greater than 20 km. Ellipsoidal earth and satellite geodetic sub-latitude exaggerated geometry are presented in Figure 3.3. Note that geodetic latitude is measured as the angle from the equatorial plane to the line normal to the surface of the earth ellipsoid, at the point in question.

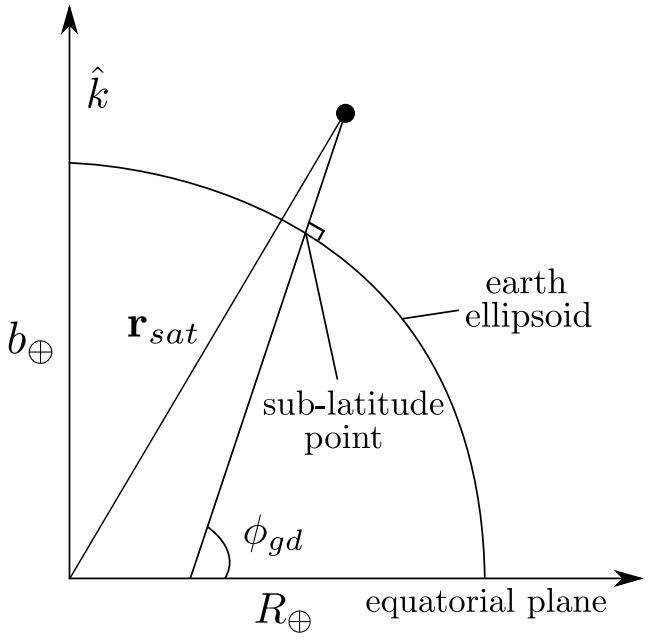

Figure 3.3: Satellite Sub-Latitude Geometry

Many ITRF to geodetic latitude and longitude conversion techniques require iteration to solve for geodetic latitude, however, Borkowski, in [29], presents an analytic solution. While implementing Borkowski's algorithm, an error was discovered in Vallado's presentation of Borkowski's algorithm, in [30], and a dialog was started with Mr. Vallado regarding a correction. Borkowski's algorithm to solve for satellite sub-longitude, λ, and geodetic sub-latitude, ϕ_{gd}, where position vector ITRF notation, r^{ITRF}, is dropped, can be summarized as:

$$
\begin{aligned}
r_{eq} &= \sqrt{r_1^2 + r_2^2} & E &= \frac{br_k - (a^2 - b^2)}{ar_{eq}} \\
\lambda &= \sin^{-1}\left(\frac{r_2}{r_{eq}}\right) & F &= \frac{br_3 + (a^2 - b^2)}{ar_{eq}} \\
P &= \frac{4(EF + 1)}{3} & Q &= 2(E^2 - F^2) \\
D &= P^3 + Q^2 & & \\
&\text{if } D > 0 & \nu &= \sqrt[3]{\sqrt{D} - Q} - \sqrt[3]{\sqrt{D} + Q} \\
&\text{if } D < 0 & \nu &= 2\sqrt{-P}\cos\left[\tfrac{1}{3}\cos^{-1}\left(\frac{Q}{P\sqrt{-P}}\right)\right] \\
G &= \tfrac{1}{2}(\sqrt{E^2 + \nu} + E) & t &= \sqrt{G^2 + \frac{F - \nu G}{2G - E}} - G \\
\phi_{gd} &= \tan^{-1}\left(\frac{a(1 - t^2)}{2bt}\right) & &
\end{aligned}
\tag{3.2}
$$

where a, is the mean equatorial radius of Earth, $R_\oplus = 6378$ km, and b is the semiminor axis of Earth, $b_\oplus = 6357$ km.

Table 3.1: Sun Elevation Angle versus Earth Illumination

Sun Elevation (degrees)	Earth Illumination (lux)
60	100,000
0	700
-6	2
-12	.01
-18	.0007

3.4 Sun Analysis

As stated above, requirements for satellite observability include that the viewing location on Earth must be in twilight while the RSO of interest must be in sunlight. Analysis to determine these two factors is presented here.

3.4.1 Earth Illumination

Vallado, in [30], presents the relationship between sun elevation angle and Earth illumination. The pertinent values for this application are shown in Table 3.1. Sun elevation angle in the local SEZ frame determines site illumination. Allowable Earth illumination values for this application are defined as less than and equal to 0.01 lux, or equivalently, a Sun elevation angle of 12 degrees or less, commonly referred to as astronomical twilight.

To determine the Sun elevation angle, the Sun position vector in GCRF must be calculated and converted to SEZ. The GCRF Sun position vector can be calculated from:

$$\mathbf{r}_{\odot} = \begin{bmatrix} r_{\odot}\cos(\lambda_{ecliptic}) \\ r_{\odot}\cos(\epsilon)\sin(\lambda_{ecliptic}) \\ r_{\odot}\sin(\epsilon)\sin(\lambda_{ecliptic}) \end{bmatrix} \tag{3.3}$$

The Astronomical Almanac, in [31], presents a simple technique to determine the Sun's distance from the center of Earth, $\mathbf{r}_{\odot}$, the obliquity of the ecliptic, ϵ, and the Sun's ecliptic longitude, $\lambda_{ecliptic}$ to give the Sun's GCRF position vector with an accuracy of 0.01 degrees. The transformation from GCRF to ITRF is presented in section 4.2.2. See Weisel for a treatment of coordinate rotation transformations [32]. The transformation from ITRF to SEZ and the subsequent sun elevation angle calculation are defined as follows:

$$\begin{aligned}
\mathbf{R}_{ITRF}^{SEZ} &= \mathbf{R}_2(\tfrac{\pi}{2} - \phi_{gd})\mathbf{R}_3(\lambda) \\
\mathbf{r}_{sun}^{SEZ} &= \mathbf{R}_{ITRF}^{SEZ}\left[\mathbf{r}_{sun}^{ITRF} - \mathbf{r}_{site}^{ITRF}\right] \\
El_{sun} &= \sin^{-1}\left(\frac{r_{3,sun}^{SEZ}}{r_{sun}^{SEZ}}\right)
\end{aligned} \tag{3.4}$$

where $\mathbf{r}_{site}^{ITRF}$ is calculated through elliptic geometry, based on site geodetic latitude, longitude, and height above the earth ellipse, assumed equal to height above mean sea level for approximation, by:

$$\begin{aligned}
C_{\oplus} &= \frac{R_{\oplus}}{\sqrt{1 - e_{\oplus}^2 \sin^2 \phi_{gd}}} & S_{\oplus} &= \frac{R_{\oplus}(1 - e_{\oplus}^2)}{\sqrt{1 - e_{\oplus}^2 \sin^2 \phi_{gd}}} \\
r_{eq} &= (C_{\oplus} + h_{ellp}) \cos \phi_{gd} & r_{3,site}^{ITRF} &= (S_{\oplus} + h_{ellp}) \sin \phi_{gd} \\
r_{1,site}^{ITRF} &= r_{eq} \cos \lambda & r_{2,site}^{ITRF} &= r_{eq} \sin \lambda
\end{aligned} \tag{3.5}$$

3.4.2 Satellite Illumination

For possible pass prediction, the Earth's partial and full shadow, eclipse regions, the penumbra and umbra respectively, are modeled as a cylinder shadow with a radius equal to Earth's mean equatorial radius and a height of infinity, starting on the sun-relative, back half of the

Earth. This is, in effect, modeling the Sun as a point mass of infinite distance from the Earth so that Sun light rays become parallel when they reach Earth. The question of if the satellite is being illuminated by the Sun reduces to a line of sight (LOS) calculation with possible Earth obstruction. If there is a LOS the satellite is illuminated. The LOS calculation is:

$$\tau_{min} = \frac{r_{sat}^2 - \mathbf{r}_{sat} \cdot \mathbf{r}_{sun}}{r_{sat}^2 + r_{sun}^2 - 2(\mathbf{r}_{sat} \cdot \mathbf{r}_{sun})}$$

$$\text{if} \quad \tau_{min} < 0 \quad \text{or} \quad \tau_{min} > 1 \qquad\qquad LOS = \text{True}$$

$$\text{else if} \quad (1 - \tau_{min}) r_{sat}^2 + (\mathbf{r}_{sat} \cdot \mathbf{r}_{sun}) \tau_{min} \geq R_{\_\oplus}{}^2 \qquad LOS = \text{True} \tag{3.6}$$

Figure 3.4 shows the Earth shadow cylinder and Sun geometry, and defines $\mathbf{r}_{sat}$ and $\mathbf{r}_{sun}$. In the figure, β is $\frac{\pi}{2}$ minus the angle from Earth's instantaneous pole, $itrf_3$, to the Sun position vector.

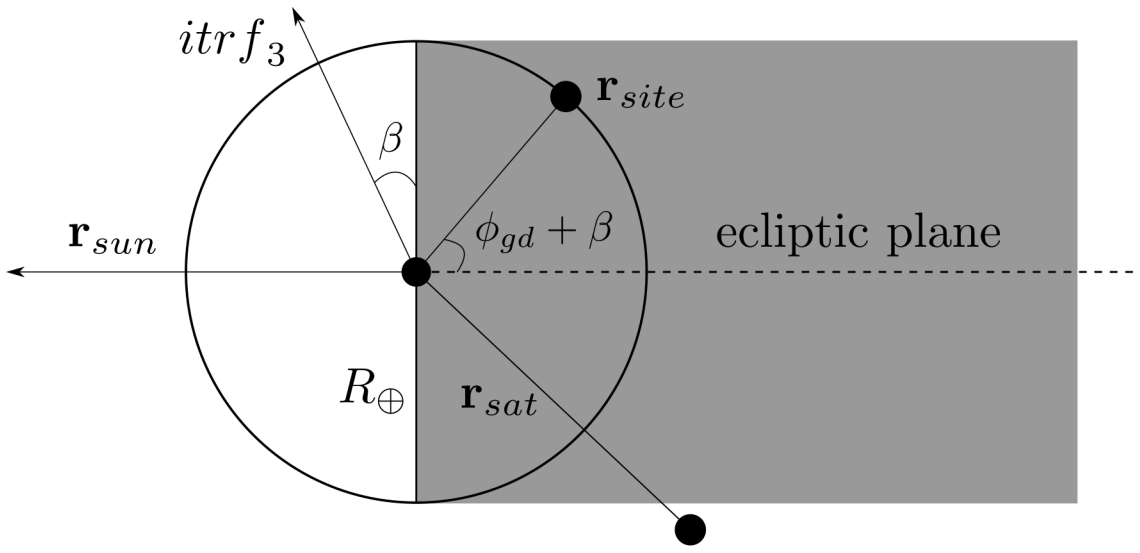

Figure 3.4: Shadow Geometry

The furthest distance from the sun of the observation site does not remain fixed with respect to the cylinder shadow. β reaches a maximum, equal to the obliquity of the ecliptic, approximately 23.4°, during the summer solstice and a minimum, equal to the negative obliquity of the ecliptic, approximately -23.4°, during the winter solstice. Although the site is in twilight for longer near the winter solstice, the minimum viewing satellite radius, or outside the shadow cylinder, is smallest near the summer solstice. Thus, more and brighter objects can be viewed near the summer solstice: in fact, near the summer solstice there exist RSOs with nominal magnitudes of less than 5 vmag that can be viewed throughout the night.

Satellite apparent magnitude is a highly complex function of satellite attitude, sun angle, material, satellite size, altitude, and other factors. A highly-accurate function is beyond the requirements of this application, as only an estimate of satellite brightness, within a few orders of magnitude is needed. Nominally, RSO magnitude increases with distance from Earth. For this application, magnitude estimates can be referenced on $heavens-above.com$ [33]. Heavens Above is a useful site for referencing possible observable satellite passes (i.e. nightly planning) and it was used to validate the algorithms in this chapter.

3.5 Visible Passes – MATLAB Function

All of the considerations in this chapter were combined into possible_sat_passes.m, a MATLAB function that predicts all passes and visible passes of a particular RSO over a user defined time interval. The output of the function is a time listing of passes and a ground track figure. A sample ground track figure is shown in Figure 3.5. The ground track is shown in black, the observable sub-latitude circle in red, and visible points of the ground track in green. This MATLAB function is in Appendix B.

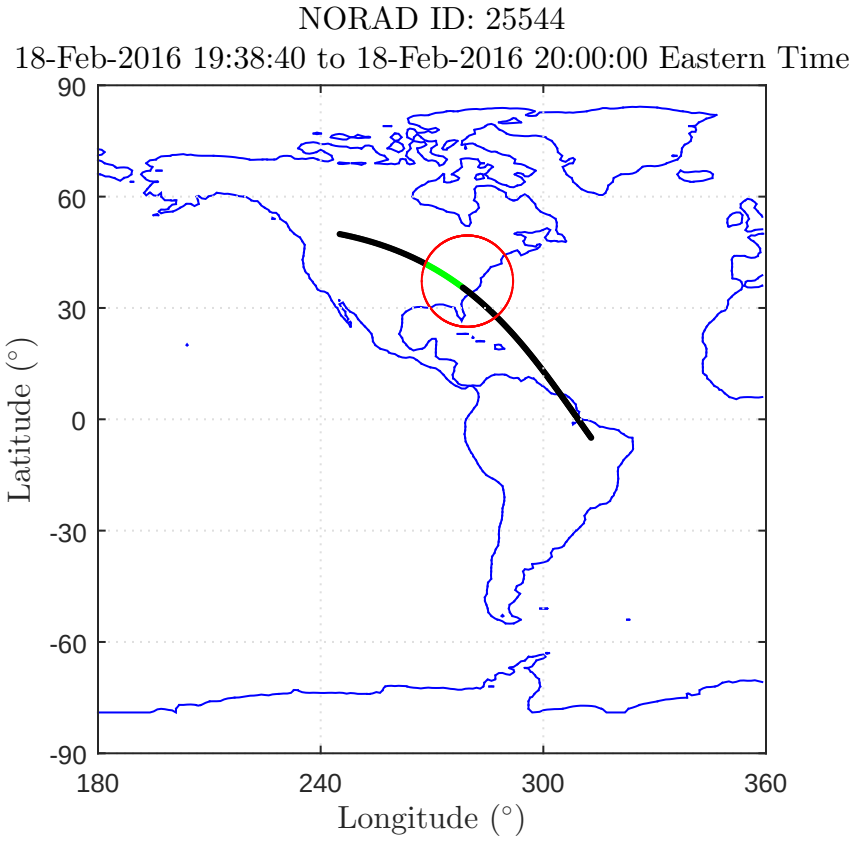

Figure 3.5: Calculated Observable Pass of the ISS

In summary, this chapter presents the reference frames, literature review, background research, and algorithms required for determining RSO observable passes. These algorithms included calculating satellite sub-latitude to determine Earth viewing geometry and Sun GCRF position analysis to determine RSO and Earth illumination. This chapter presents the information needed to plan nightly telescope tasking and the viewing geometry and reference frame transformations that will be used in subsequent derivations.

Chapter 4

Surveillance

Surveillance is defined as observing the celestial sphere either where you expect an RSO to be, based on orbital propagation, or a general location to see if any RSOs pass through during the observational period. This is an open-loop problem, i.e. the RSO's position in the image is not determined real-time. This chapter will present the background information and algorithms used in VTOST surveillance procedures.

4.1 Orbital Propagation – SGP4

The orbital propagation analyzed and used for VTOST is the Air Force Space Command (AFSPC) Simplified General Perturbations 4 (SGP4) model, summarized by Hoots, Schumacher, and Glover in [34]. SGP4 is an analytic model based on Brouwer's theory, presented in [35], with an added drag effect analysis. The mean orbital elements used in SGP4, however, take the form presented by Kozai, in [36]. For real-world applications the definition of "mean" elements must be clearly understood.

SGP4 accounts for the Earth gravitational field using only zonal harmonics. Secular and long periodic effects up to J_4 are considered, however, only second degree short periodic perturbations are calculated. A power density function is used to model a non-rotating, spherical atmospheric density for drag considerations [37]. q_0 and s are constant parameters of the SGP4 density function. Third body, solar pressure, and tidal and tectonic motion effects are not modeled.

SGP4 uses information from a two-line element set (TLE) representing the epoch state. TLEs are generated for most man-made satellites for public use by JFCC Space. The information is available through *space-track.org*, once an account is created. A MATLAB function was generated to look up the most recent TLE for a satellite based on its North American Aerospace Defense Command (NORAD) catalog identification number. The function is presented in Appendix B.

One limitation of SGP4 is the lack of uncertainty estimates for either propagated states or the TLE epoch state. This is not a major issue for surveillance where a specific location is observed, regardless of if a satellite is located there or not. For tracking, however, estimating the propagation uncertainty becomes crucial. The main factor affecting accuracy of the propagation is time from TLE epoch [38]. Since SGP4 does not have uncertainty estimates, numerous specific validation cases must be investigated to determine accurate, general SGP4 uncertainty estimations. Based on validation studies by Vallado [39], Kelso [40], Greene [41], and Oltrogge [42] general SGP4 position and velocity uncertainty estimates have been approximated as functions of time from TLE epoch for use in Chapter 6.

The SGP4 method can be summarized as follows: The Kozai mean orbital elements and other orbital parameters at epoch, SGP4 mean motion n_0, eccentricity e_0, inclination i_0, mean anomaly M_0, argument of perigee ω_0, right ascension of ascending node Ω_0, time rate change of the mean motion $\dot{n}_0$, and SGP4 drag coefficient B^* are given through TLE lookup,

and $t - t_0$ is the time since TLE epoch. Perturbations are then calculated and added to the mean elements to find the osculating elements.

True Equator Mean Equinox (TEME) unit vectors, where the osculating elements, Ω_k, i_k, u_k, r_k, $\dot{r}_k$, and $\dot{f}_k$, are calculated using the SGP4 algorithm presented in Appendix D, are calculated as:

$$\mathbf{M} = \begin{bmatrix} -\sin\Omega_k \cos i_k \\ \cos\Omega_k \cos i_k \\ \sin i_k \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} \cos\Omega_k \\ \sin\Omega_k \\ 0 \end{bmatrix}$$

$$\mathbf{U} = \mathbf{M}\sin u_k + \mathbf{N}\cos u_k$$

$$\mathbf{V} = \mathbf{M}\cos u_k - \mathbf{N}\sin u_k \tag{4.1}$$

The position and velocity vectors are then calculated in the TEME frame as:

$$\mathbf{r}^{TEME} = r_k\mathbf{U}$$

$$\mathbf{v}^{TEME} = \dot{r}_k\mathbf{U} + r\dot{f}_k\mathbf{V} \tag{4.2}$$

A MATLAB listing of SGP4 is publicly available as an accompaniment to [30]. This code was verified, validated, and altered to increase efficiency for this application.

SGP4 generates ephemerides in an intermediate, inertial, True Equator Mean Equinox (TEME) frame [43]. However, all analyses for this application are done in the GCRF:

therefore a TEME to GCRF transformation must be developed. A method to perform a transformation from TEME to the previous IAU inertial frame is provided by [44], however, a method that transforms TEME to GCRF needed to be developed for this project. The MATLAB function is presented in Appendix B and the algorithm is summarized as:

$$\begin{aligned}\mathbf{r}^{GCRF} &= \mathbf{R}_{PN}(t)\mathbf{R}_3(-Eq_{equinox2000})\mathbf{r}^{TEME} \\ \mathbf{v}^{GCRF} &= \mathbf{R}_{PN}(t)\mathbf{R}_3(-Eq_{equinox2000})\mathbf{v}^{TEME}\end{aligned} \tag{4.3}$$

where $\mathbf{R}_{PN}$ is developed in subsection 4.2.2 and $Eq_{equinox2000}$ is the equation of the equinoxes calculated from the classic approach of the IAU-2000 Nutation Resolutions, presented in [30].

SGP4 propagates based on initialization at the TLE epoch, however, for this application a sequential based SGP4 was developed so that the epoch could be reinitialized when measurement updates are incorporated. This was necessary because the B^* value provided by the TLE is difficult to accurately propagate without RSO attitude knowledge. This development of a new SGP4 for a sequential, tracking filter application is presented below. The general approach is that drag is treated as a perturbation that can be estimated using the original TLE epoch and nominal propagation. Consider that a measurement update to the position is processed and a new SGP4 epoch at t_k is required to propagate the state to t_k. Note that the TEME reference frame notation is dropped in this derivation. The proposed technique is to start the nominal propagation from TLE epoch, t_{epoch}, to t_k and t_{k+1}, represented as:

$$\begin{aligned}\mathbf{n}_k &= SGP4(\mathbf{x}_0, t_k - t_{epoch}) \\ \mathbf{n}_{k+1}^{drag} &= SGP4(\mathbf{x}_0, t_{k+1} - t_{epoch}) \\ \Delta_{drag}^{nom} &= \mathbf{n}_{k+1}^{drag} - \mathbf{n}_k\end{aligned} \tag{4.4}$$

where n represents the nominal state vector and $SGP4$ represents the nominal SGP4 propagation function for which t_{epoch} is equal to t_0. Now $SGP4^*$ will be used to denote the updated epoch, no-drag SGP4 function with t_{epoch} equal to t_k and B^* set to zero.

$$\mathbf{n}_{k+1}^{no-drag} = SGP4^*(\mathbf{n}_k, t_{k+1} - t_k)$$

$$\Delta_{no-drag}^{nom} = \mathbf{n}_{k+1}^{no-drag} - \mathbf{n}_k \tag{4.5}$$

The true drag perturbation is approximated as:

$$\Delta\mathbf{x}_{drag} \approx \Delta\mathbf{n}_{drag} = \Delta_{drag}^{nom} - \Delta_{no-drag}^{nom} \tag{4.6}$$

and the new state is calculated as:

$$\mathbf{x}_{k+1} = SGP4^*(\mathbf{x}_k, t_{k+1} - t_k) + \Delta\mathbf{x}_{drag} \tag{4.7}$$

4.2 Transforming Celestial Coordinates to Terrestrial Coordinates

With the satellite GCRF state known from SGP4 as a function of time, the state in the topocentric celestial reference frame (TCRF) and SEZ must be determined. TCRF states are used to task the telescope and SEZ states are used to determine local pass parameters.

4.2.1 GCRF to TCRF Transformation

When viewing a satellite from anywhere except the center of the Earth, Geocentric parallax must be considered. Geocentric parallax can be thought of as the apparent right ascension and declination of an RSO. Therefore, to view an RSO, a mount must be tasked with the RSO's apparent, or topocentric right ascension and declination rather than its true right ascension and declination.

Figure 4.1 shows the difference between declination, δ and topocentric declination, δ_t: the same principle applies for right ascension. For stars, the large magnitude difference between $\mathbf{r}_{site}$ and $\mathbf{r}_{star}$ make the difference between geocentric and topocentric angles negligible.

The transformation from right ascension and declination to topocentric right ascension and

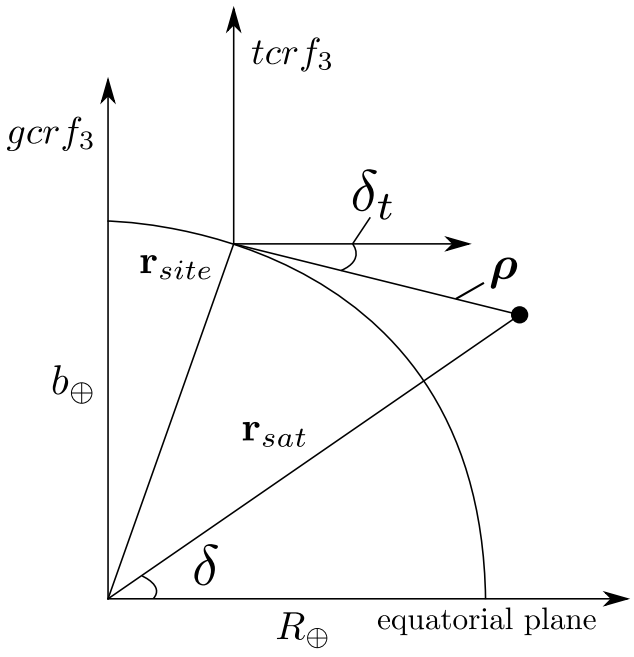

Figure 4.1: Geocentric Parallax Geometry

declination, and associated rates, can be calculated as:

$$\begin{aligned}
\boldsymbol{\rho} &= \boldsymbol{\rho}^{GCRF} = \mathbf{r}_{sat}^{GCRF} - \mathbf{R}_{ITRF}^{GCRF}\mathbf{r}_{site}^{ITRF} & \dot{\boldsymbol{\rho}} &= \dot{\boldsymbol{\rho}}^{GCRF} = \mathbf{v}_{sat}^{GCRF} - \mathbf{v}_{site}^{GCRF} \\
\delta_t &= \sin^{-1}\left(\frac{\boldsymbol{\rho}_3}{\rho}\right) & \alpha_t &= \cos^{-1}\left(\frac{\rho_1}{\sqrt{\rho_1^2+\rho_2^2}}\right) \\
\dot{\rho} &= \frac{\boldsymbol{\rho}\cdot\dot{\boldsymbol{\rho}}}{\rho} & \dot{\alpha}_t &= \frac{\dot{\rho}_1\rho_2 - \dot{\rho}_2\rho_1}{-\rho_2^2-\rho_1^2} \\
\dot{\delta}_t &= \frac{\dot{\rho}_3 - \dot{\rho}\sin\delta_t}{\sqrt{\rho_1^2+\rho_2^2}}
\end{aligned} \tag{4.8}$$

where $\mathbf{r}_{site}^{ITRF}$ can be found from site geodetic latitude and longitude, as shown in equation 3.5. However, $\mathbf{R}_{ITRF}^{GCRF}$ and $\mathbf{v}_{site}^{GCRF}$ still need to be determined. The next section shows the GCRF to ITRF transformation. Note that there is a singularity when the RSO is at zenith, however, there are also imaging issues with the dome near zenith so checks in the software will be used to avoid this singularity.

4.2.2 GCRF to ITRF Transformation

The GCRF to ITRF frame reduction formula can be defined as:

$$\begin{aligned}
\mathbf{r}^{GCRF} &= \mathbf{R}_{PN}(t)\mathbf{R}_R(t)\mathbf{R}_W(t)\mathbf{r}^{ITRF} \\
\mathbf{v}^{GCRF} &= \mathbf{R}_{PN}(t)\mathbf{R}_R(t)[\mathbf{R}_W(t)\mathbf{v}^{ITRF} + \boldsymbol{\omega}_\oplus \times \mathbf{R}_W(t)\mathbf{r}^{ITRF}]
\end{aligned} \tag{4.9}$$

where $\mathbf{R}_{PN}$ is the precession-nutation rotation matrix of date, $\mathbf{R}_R$ is the sidereal rotation matrix of date, and $\mathbf{R}_W$ is the polar motion rotation matrix of date [30]. Note that the time dependence notation will be dropped in this section for conciseness.

The formulation technique of these rotation matrices is an often studied problem and the

prevailing technique is updated when accuracy is increased. The current approach, adopted 1 January 2009, is defined by the IAU-2006/2000 Resolutions. For completeness, the reduction formulation used in this application uses the IAU-2006 theory of precession and the IAU-2000 theory of nutation with the Celestial Intermediate Origin (CIO) approach as defined by the IAU. MATLAB functions provided by Vallado to implement this reduction have been verified and altered to increase speed. The reduction formulation is extensive: a pertinent summary as needed to formulate the measurement model is presented below. A full derivation is presented in [30]. The polar motion rotation matrix is be defined as:

$$\mathbf{R}_W = \mathbf{R}_3(-s')\mathbf{R}_2(x_p)\mathbf{R}_1(y_p) \tag{4.10}$$

where s' is the time dependent Terrestrial Intermediate Origin Locator to account for the instantaneous prime meridian and x_p and y_p are the time dependent polar motion angles, to account for polar motion. The National Geospacial-Intelligence Agency (NGA) publishes weekly polynomial coefficients that can be used to estimate instantaneousness polar motion angles. MATLAB functions were created to lookup the NGA coefficients and linearly interpolate the polar motion angles for any time. These functions are presented in Appendix B. The sidereal rotation matrix is defined as:

$$\mathbf{R}_R = \mathbf{R}_3(-\theta_{ERA}) \tag{4.11}$$

where θ_{ERA} is the time dependent Earth Rotation Angle, which can be thought of as similar to Greenwich Sidereal Time. Finally, the precession-nutation rotation matrix is defined as:

$$\mathbf{R}_{PN} = \begin{bmatrix} 1-aX^2 & -aXY & X \\ -aXY & 1-aY^2 & Y \\ -X & -Y & 1-a(X^2+Y^2) \end{bmatrix} \mathbf{R}_3(s) \tag{4.12}$$

where s, a, X, and Y are calculated based on known time dependent functions. Precession and nutation functions are dominated by gravitational effects of the Sun, Moon, and other planets. The U.S. Naval Observatory (USNO) publishes daily Earth Orientation Parameters (EOP) that contain dX and dY, time dependent correction factors. MATLAB functions were created to lookup the USNO daily predictions and linearly interpolate the corrections for any time. These functions are presented in Appendix B. To calculate the final form of the reduction, flip the rotation by reversing the order of the rotation sequence and taking the transpose of each rotation matrix. The final result is:

$$\begin{aligned} \mathbf{r}^{ITRF} &= \mathbf{R}_W^T \mathbf{R}_R^T \mathbf{R}_{PN}^T \mathbf{r}^{GCRF} \\ \mathbf{v}^{ITRF} &= \mathbf{R}_W^T [\mathbf{R}_R^T \mathbf{R}_{PN}^T \mathbf{v}^{GCRF} - \boldsymbol{\omega}_\oplus \times \mathbf{R}_R^T \mathbf{R}_{PN}^T \mathbf{r}^{GCRF}] \end{aligned} \tag{4.13}$$

Now the required rotations to convert from GCRF to TCRF are known. In order to calculate local angles, azimuth and elevation, and their respective rates an additional rotation must be calculated. This transformation is presented in the next section.

4.2.3 Transforming ITRF to SEZ Coordinates

Once in ITRF the calculation of azimuth and elevation becomes a subtraction of the ITRF site position vector to move the frame origin to the site; a rotation to transform from ITRF to SEZ; and a Cartesian to spherical reduction as azimuth and elevation are the spherical angle variables in the SEZ frame. In the Cartesian to spherical reduction the SEZ frame denotation for position and velocity vectors is dropped.

$$\mathbf{R}_{ITRF}^{SEZ} = \mathbf{R}_2(\tfrac{\pi}{2} - \phi_{gd})\mathbf{R}_3(\lambda)$$

$$
\begin{aligned}
\mathbf{r}^{SEZ} &= \mathbf{R}^{SEZ}_{ITRF}(\mathbf{r}^{ITRF} - \mathbf{r}^{ITRF}_{site}) \\
\mathbf{v}^{SEZ} &= \mathbf{R}^{SEZ}_{ITRF}\mathbf{v}^{ITRF} \\
az &= \tan^{-1}\left(\frac{\mathrm{r}_2}{\mathrm{r}_1}\right) \\
el &= \sin^{-1}\left(\frac{\mathrm{r}_3}{\mathrm{r}}\right) \\
\dot{az} &= \frac{\mathrm{v}_1\mathrm{r}_2 - \mathrm{r}_1\mathrm{v}_2}{\mathrm{r}_1^2 + \mathrm{r}_2^2} \\
\dot{el} &= \frac{\mathrm{v}_3(\mathrm{r}_1^2 + \mathrm{r}_2^2) - \mathrm{r}_3(\mathrm{r}_1\mathrm{v}_1 + \mathrm{r}_2\mathrm{v}_2)}{\mathrm{r}^2\sqrt{\mathrm{r}_1^2 + \mathrm{r}_2^2}}
\end{aligned}
\tag{4.14}
$$

4.2.4 Observation Phenomena

When optically observing an RSO from a location on Earth's surface, atmospheric and other effects must be considered for highly accurate systems. These phenomena include geocentric parallax, atmospheric refraction, diurnal and annual aberration, relativistic deflection effects, and light time. As stated above, the largest effect of the mentioned phenomena is geocentric parallax, which has already been accounted for. Atmospheric refraction causes objects to appear at higher than the true elevation. Holland, in [45], has shown that refraction value for LEO satellites reaches a maximum of about 2" when observing near the site horizon. For comparison purposes, mapped to a 400 km circular orbit, 1" is about equal to 2 meters.

Annual aberration, or the effect on light of the earth orbiting around the sun, is a phenomenon that has been long studied due to its effect on stellar observations. Doolittle states that the maximum value, or constant of annual aberration is about 20.5" [46]. Lang, in

[47], shows that the maximum value for diurnal aberration, or the effect on light due to the rotation of the earth, to be 0.32". A preliminary light-time correction calculation, for a 400 km circular orbit, is about 5.3", is given by

$$\Delta = \frac{v_{orbit}}{c} \text{ rad} \tag{4.15}$$

where c is the speed of light.

Vallado states that relativistic light deflection for near-Earth satellites is around 1 cm and thus, will not be considered [30]. The complexity of accounting for these phenomena is not worth their implementation, as SGP4 orbit propagation errors will dominate the uncertainty for this surveillance application. These phenomena must be modeled for accurate OD; however, for this surveillance application all that is needed is for the predicted RSO position to be in the FOV of the camera, which allows an error tolerance of about 200'.

4.3 Results – Surveillance Operational Code

With the satellite state in GCRF, TCRF, and SEZ known at each timestep, the minimum range, from the satellite to the site, and its respective time can be calculated. This calculation is based on TLE and pass time range information, both detailed in previous chapters, and input by the operator for each RSO. User input is written into nightly_tasking_user_input.m which is presented, with sample pass information, in Appendix B. The minimum range and the TCRF and SEZ state calculations are performed in surveillance_function.m presented in Appendix B. Note that the pointing information is saved for image post processing in the "collected_data" subfolder of "telescope control" with the naming convention "yy_mm_dd_pointing_data" where the date listed is the operation date.

This information is then used in the surveillance operational code. The operational code automatically tasks the mount and camera to operate at the necessary times, i.e. a full night of tasking will require no user operation. This code is presented in Appendix B and a full operational procedure listing, including hardware setup, for surveillance is listed in Appendix C. The surveillance code tasks the mount and camera using the tasking function presented in Appendix B. Images are taken for a minute surrounding the time of minimum range and are saved in the "collected_data" subfolder of "telescope control" with the naming convention "norad_id_yy_mm_dd_hh_mm_sssss" where the time listed is the end of the exposure for that image.

The results presented in this chapter are the full surveillance implementation for VTOST. Operational procedures and code were verified by performing multiple nights of data collection. Figure 4.2 shows surveillance pass data of the ISS, collected on April 12th, 2016. The lower left in the images is near the horizon, and does not have many visible stars. The images are sequential from upper left to lower right.

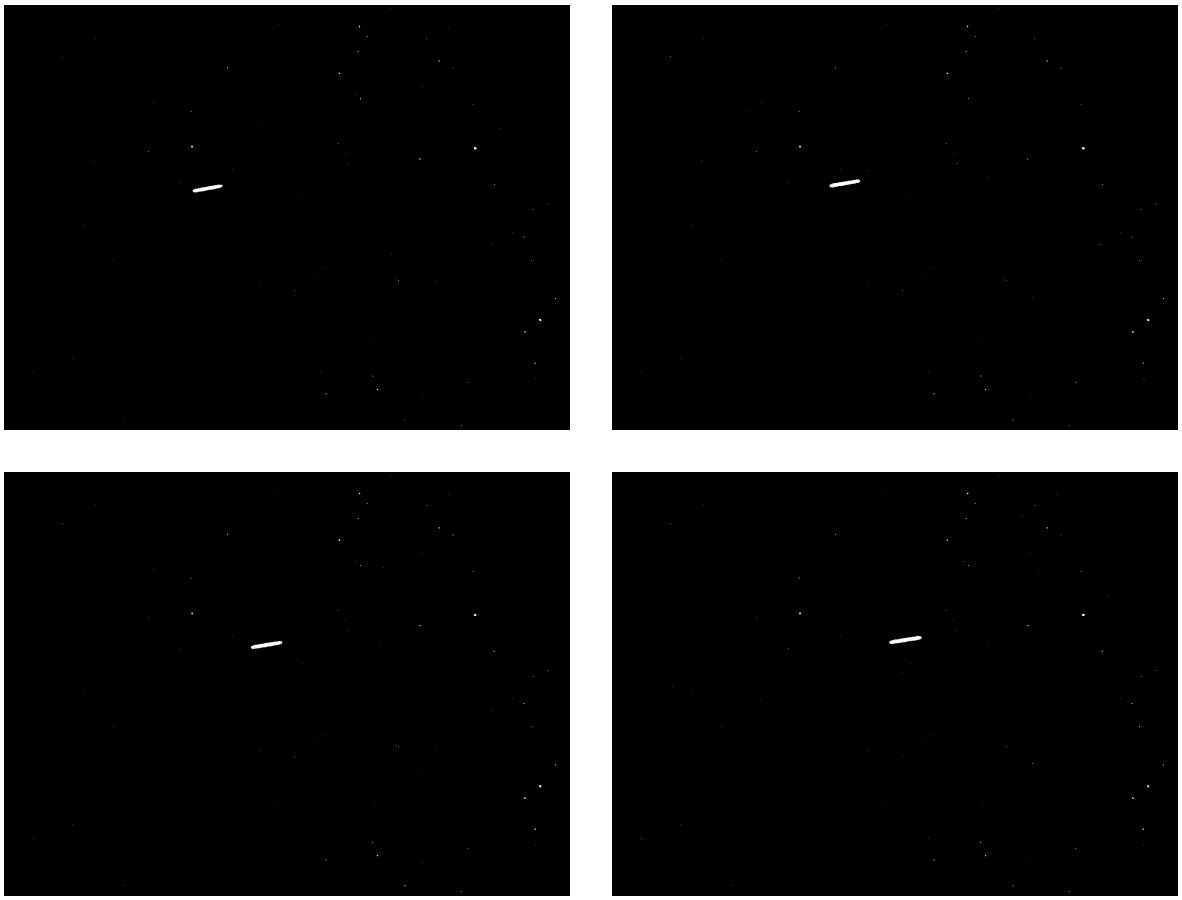

Figure 4.2: Surveillance Pass Data (sequential from upper left to lower right)

Chapter 5

Tracking – Body Frame to TCRF Method

5.1 Image Processing – GEODETICA

There has been recent research on RSO image streak analysis, including detection within a known astronomical catalog [48] and object classification algorithms [49]. Sease and Flewelling, in [50], present GEODETICA, "a software platform for detection, tracking, and discrimination of objects in continuous sequences of unresolved space-based imagery". It will be shown that this software platform can be utilized for VTOST if the viewing geometry is analyzed and appropriate transformations are developed.

GEODETICA stands for the General Electro-Optical Detection, Tracking, Identification, and Characterization Application tool. The tool uses a Kalman Filter multi-hypothesis bank for point tracking along with edge detection and phase congruency and does not depend on an astronomical catalog. GEODETICA processes continuous unresolved imagery and produces

image plane motion and light curve data. A full presentation of GEODETICA is given by Sease in [51].

5.1.1 Astrometry.net

Astrometry.net is a software package, developed by Hogg, et al. [52], that performs an image astrometric reduction using geometric hashing. Basically, this software determines the stars in the image, the field of view of the image, the pixel per real-world degree, and other image calibration information. Sease has created code that allows Astrometry.net code to run in tandem to GEODETICA, allowing for additional measurement updates to increase accuracy. This thesis lays the groundwork for a full tracking system that employs both GEODETICA and Astrometry.net software in real-time. Procedures for installing and using Sease's GEODETICA platform and Astrometry.net MATLAB wrapper are presented in Appendix A.

5.2 Camera Body Reference Frame Analysis

5.2.1 Transformation from Image Frame to Body Frame

Since the GEODETICA derivation and output is in the image reference frame, the map from the image frame, IM, to the body frame, B, must be developed. Time-dependent notation is dropped for conciseness. In state space this is represented as:

$$\mathbf{x} = [r_1, r_2, r_3, \dot{r}_1, \dot{r}_2, \dot{r}_3]^T$$

$$\mathbf{x}^B = \mathbf{h}(\mathbf{x}^{IM}) \tag{5.1}$$

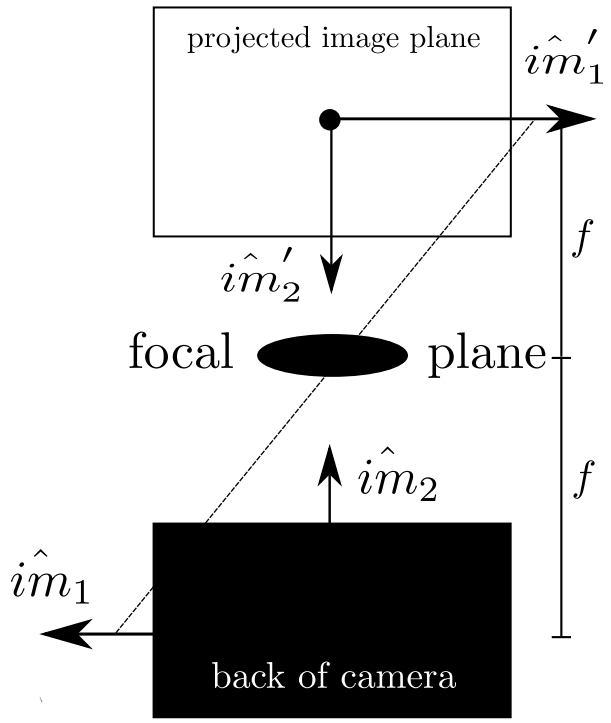

Figure 5.1: Pin-hole Camera Model

The map is found using the pin-hole camera model, presented in [53], and shown in Figure 5.1. The pin-hole model is an approximation of a rectilinear projection using an affine transformation, involving a reflection and translation. Basically, the pin-hole model represents how light is reflected across the focal plane. This approximation assumes a perfect lens and no spherical distortion, an assumption often made when considering small FOV star sensors [54]. Most importantly for the subsequent derivation is that affine transformations preserve collinearity.

The pin-hole camera model maps all points in the image plane to the celestial sphere projected image plane. In other words, the image plane is projected through the focal plane and when this happens the image plane is reflected. The origin of the projected image reference frame, IM', is translated a focal length distance from the focal point, along the boresight.

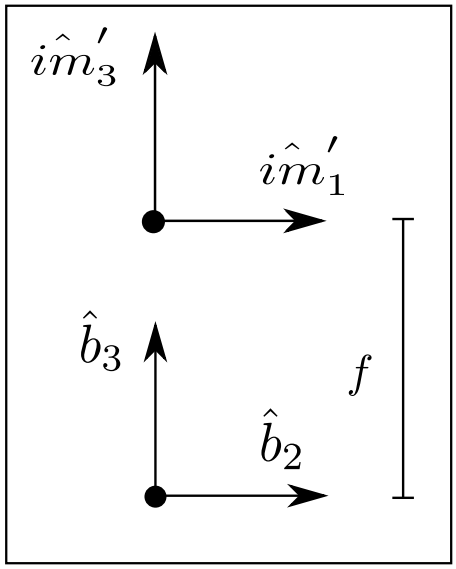

Figure 5.2: Body Frame and Projected Image Frame, Looking Down on Camera

Analyzing Figure 5.1 shows that the pin-hole model affine transformation of the image plane is a rotation of the original image reference frame, IM, around the boresight of π radians and a translation along the boresight of $2fi\hat{m}_3$, where f is the focal length. Noting that the motion is constrained to the $i\hat{m}_1 - i\hat{m}_2$ plane and that the reflected image plane and body plane are both body fixed, i.e. do not rotate with respect to each other, simplifies the velocity transformation. In matrix notation:

$$\begin{aligned} \mathbf{r}^{IM'} &= \mathbf{R}_3(\pi)\mathbf{r}^{IM} + 2fi\hat{m}_3' \\ \mathbf{v}^{IM'} &= \mathbf{R}_3(\pi)\mathbf{v}^{IM} \end{aligned} \tag{5.2}$$

The body frame, centered at the focal plane, is now defined in Figure 5.2.

To complete the map, consider Figures 5.1 and 5.2 and the fact that the origin of the body frame is at the focal plane. The transformation from the projected image frame to the body

frame can then be calculated as:

$$\mathbf{r}^B = \mathbf{R}_3(-\tfrac{\pi}{2})\mathbf{r}^{IM'} - f\hat{b}_3$$

$$\mathbf{v}^B = \mathbf{R}_3(-\tfrac{\pi}{2})\mathbf{v}^{IM'} \tag{5.3}$$

5.2.2 Body Frame Dynamics

To employ GEODETICA for this application, a continuous model of the satellite dynamics, in the body frame designated by superscript B, must be developed. Sease et al. [55], have shown that satellite motion in the image plane over a short timespan is well modeled by a linear function. This section will derive the linear dynamic model in the body frame, based on linear motion in the image plane. Although the affine transformation described in the section above is nonlinear due to the translation, the transformation from $\hat{im}_1$ and $\hat{im}_2$ to $\hat{b}_1$ and $\hat{b}_2$ is linear. This is due to the fact that rotations and translations in the affine transformation are all with respect to the $\hat{im}_3 = \hat{b}_3$ axis. Thus, linear motion in the $\hat{im}_1 - \hat{im}_2$ plane will result in linear motion in the $\hat{b}_1 - \hat{b}_2$ plane after transformation.

The unit vector for a point on the image plane in the body frame, where r_1^B and r_2^B are found from the transformation described above, is:

$$\hat{\mathbf{r}}^B = \frac{1}{\sqrt{r_1^2 + r_2^2 + f^2}} \begin{bmatrix} r_1 \\ r_2 \\ f \end{bmatrix} \tag{5.4}$$

Since motion is constrained to the $\hat{b}_1 - \hat{b}_2$ plane due to $\dot{f} = 0$, the dynamics can be developed

in two dimensions. The linear dynamics, in the body frame, are derived as:

$$\mathbf{x} = \begin{bmatrix} r_1 \\ r_2 \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

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

$$\mathbf{\Phi}|_{\Delta t} = \mathrm{e}^{\mathbf{A}\Delta t} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Delta t = t_{k+1} - t_k$$

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k \tag{5.5}$$

The linear body model will govern the motion until a measurement update is available from GEODETICA. Using the measurement, the state vector for that timestep is updated, in effect re-linearizing the motion in the body frame around the measurement for propagation over the next timespan. The full dynamic model with incorporation of GEODETICA updates is shown in Figure 5.3, where h is the transformation function from the GEODETICA image frame to the body frame, derived in the previous section. Updates from GEODETICA are

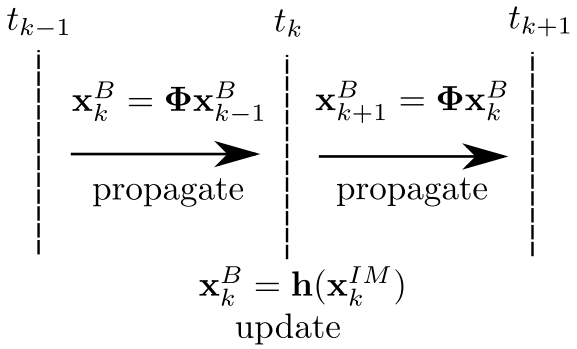

Figure 5.3: Dynamic Model in the Body Frame

necessary over the pass, due to the changing RSO velocity relative to the site, i.e., the RSO will appear to move the fastest during its closest approach, or maximum elevation.

5.3 Transformation from Body Frame to TCRF

To task the telescope from the resulting body dynamics, the satellite's topocentric celestial coordinates must be determined from the body frame unit vector. This section shows these coordinates can be found by mapping the body frame to the TCRF by considering the spherical celestial geometry of the problem. Others, including Singla in [54], use a similar technique for attitude determination, however, their techniques require astrometric reductions which involves knowing star positions in the GCRF. This new technique does not require astrometric reductions; it was developed for this application based on the specific celestial geometry involved.

The starting point for this technique is to assume there is no stellar parallax. This is an often used assumption due to the massive GCRF radii of stars. In other words, for stars, the difference between topocentric right ascension and declination and right ascension and declination is negligible. This, in effect, is equivalent to collapsing the earth to a point mass.

Thus, with respect to the stars, the observation site is now at the center of the Earth and, importantly, the GCRF and body frame now have the same origin. In turn, celestial sphere geometry can be considered where topocentric and geocentric celestial coordinates are equal. The celestial sphere can be pictured using Figure 3.1 where all of the stars are projected onto the sphere, with a radius set to one.

An important realization, however, is that any analysis in this system done on RSOs will still be in topocentric right ascension and declination. This is exactly what is needed to task the telescope so that no further reductions must be made. Furthermore, converting from geocentric to topocentric celestial coordinates, for the RSO, is not possible because information about RSO range is unobservable from image plane analysis. Figure 5.4 shows the site-centered observable geometry of this system.

Note that the great circle connecting the north horizon point, the celestial sphere north pole, and zenith is referred to as the local meridian, or meridian, and the right ascension of the meridian is known as local sidereal time (LST).

Now, in this system, one of the telescope's axes of rotation is aligned with the Earth's axis

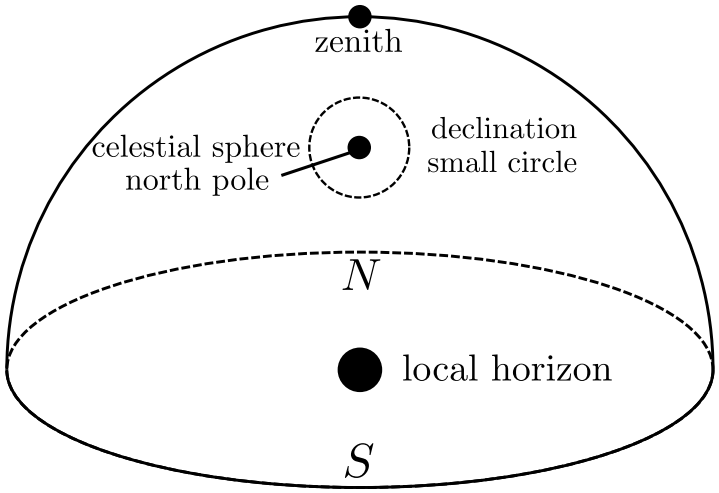

Figure 5.4: Site-Centered Observable Hemisphere

of rotation (i.e. the axis of rotation points at the celestial north pole). Although both the TCRF and body frame are centered at the observatory site, insight is found when looking at the celestial sphere projected geometry. The telescope is set up such that when it is pointing along the meridian, the projected body frame's $\hat{b}_1$ axis is parallel with the meridian. Any rotation about the north pole results in the $\hat{b}_1$ axis remaining normal to the declination small circle. This coordinate system is described by Barry, in [56]. The origin of the body frame in this system, in right ascension and declination, is known from tasking analysis and is designated $(\alpha_{BO}, \delta_{BO})$. Figure 5.5 shows this geometry, zoomed into the declination small circle, or the small circle of constant declination.

Note that when viewing any point on the celestial sphere below the great circle that intersects the east horizon point, celestial north pole, and west horizon point, shown in Figure 5.6 as the horizontal radius, the $\hat{b}_1$ axis points towards the celestial north pole, rather than away from it. This allows the $\hat{b}_1$ axis to remain normal to the declination small circle without the

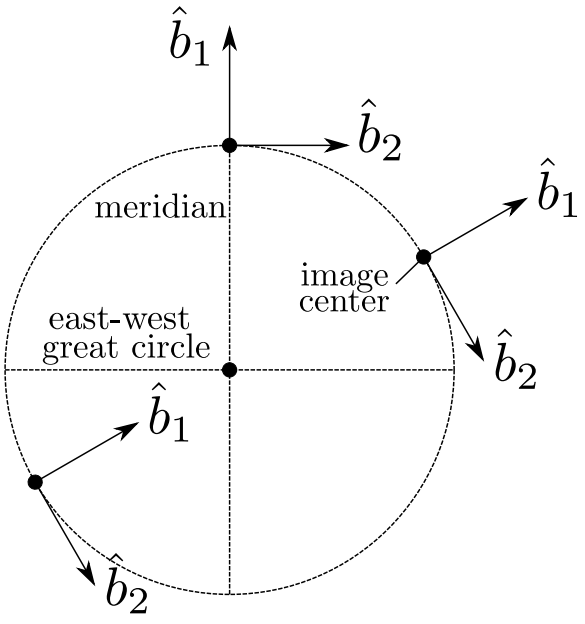

Figure 5.5: North Pole Projected Declination Small Circle and Body Frame Geometry

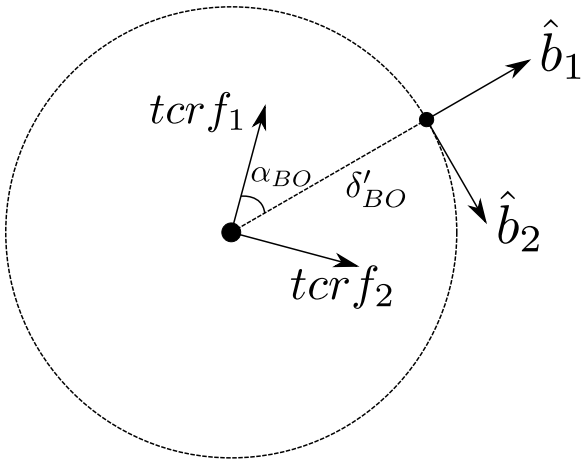

Figure 5.6: North Pole Projected TCRF and Body Frame Geometry

telescope having to turn upside down.

Figure 5.6 shows north pole projected TCRF and body frame. Analysis of this figure leads to the rotation sequence for the transformation from body frame to TCRF. Note that at the north pole declination is equal to $\frac{\pi}{2}$ and that the TCRF frame does not remain fixed with respect to the meridian, as it is rotating about negative $tcrf_3$ at Earth's rotational rate. The transformation is derived as:

$$\begin{aligned}
\mathbf{R}_B^{TCRF} &= \mathbf{R}_3(-\alpha_{BO})\mathbf{R}_2(-(\tfrac{\pi}{2} - \delta_{BO})) \\
\mathbf{r}^{TCRF} &= \mathbf{R}_B^{TCRF}\mathbf{r}^B \\
\alpha_t &= \tan^{-1}\left(\frac{r_2}{r_1}\right) \\
\delta_t &= \tan^{-1}\left(\frac{r_3}{\sqrt{r_1^2 + r_2^2}}\right)
\end{aligned} \tag{5.6}$$

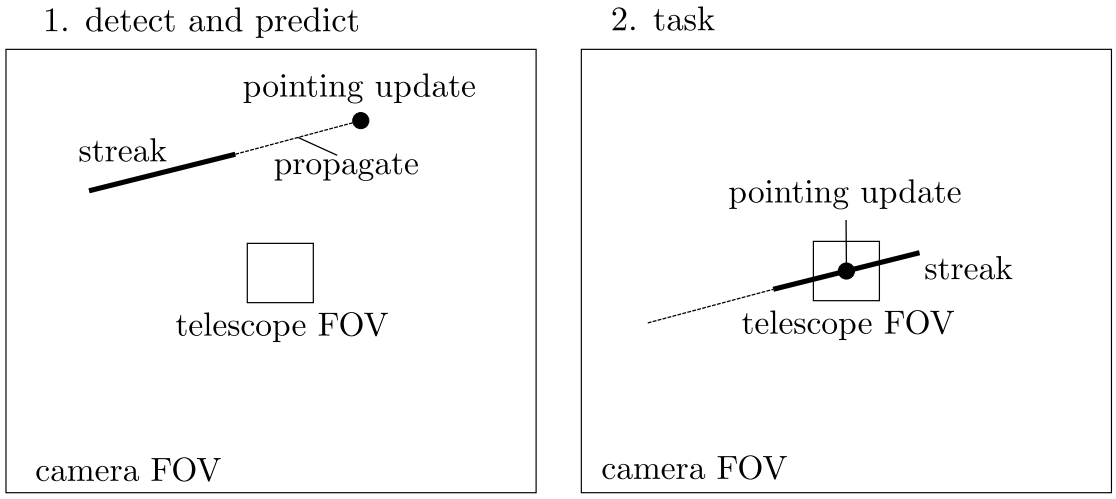

Figure 5.7: Pointing Update Schematic

5.4 Pointing Update

Once the RSO streaks have been processed by GEODETICA and the image plane position and rate is determined, the body frame dynamics can be used to calculate a future position of the RSO. Accurate velocity control is unavailable for the current telescope: if tasked to a specific right ascension and declination it will slew near the tasked point quickly and have small tuning settling time. Therefore, a pointing update is proposed that will allow the telescope to recenter the RSO in the image plane, in effect a large discrete timestep update. Figure 5.7 represents the pointing update scheme. With the uncertainty in SGP4 propagation, the probability is low that the RSO will be in the FOV of the telescope. Thus, this detection and pointing update will allow the telescope to view the RSO. This pointing update is being implemented for VTOST, which will allow for future tracking work that will aim to increase pass coverage by decreasing the propagation timestep and increasing the number of pointing updates. While not having full mount velocity control, this chapter has presented a closed-loop tracking control solution for the current hardware setup and has laid the groundwork for tracking implementation with full velocity control.

Chapter 6

Tracking – Statistical Orbit Determination

In addition to the image plane motion analysis presented in the previous chapter, tracking accuracy may be increased by including additional information from statistical orbit determination. In effect, the orbital dynamics become the RSO dynamic model rather than linear motion in the image plane. To track an RSO, a filter must be developed to synthesize dynamic model propagation and image plane measurements to calculate the best state estimate at a given time.

6.1 Dynamic Model

The dynamic model used for this filter derivation is the altered SGP4 method presented in section 4.1. Instead of increasing the complexity of modified SGP4 to account for numerous perturbation effects, the process is considered stochastic and zero-mean, white, and Gaussian noise is introduced. A random variable, x, is considered white noise if $x(t_1)$ is independent

from $x(t_2)$ for all $t_1 \neq t_2$. Modeling the noise as Gaussian is commonly accepted in practice due to the Central Limit Theorem. Oppenheim, in [57], proves that the sum of a sequence of zero-mean random variables of unknown probability distribution converges to a Gaussian probability density function (pdf). Henceforth, the following notation for a zero-mean Gaussian random variable, x, with a variance of σ^2, and its corresponding pdf, $p_x(x)$, will be used:

$$x \sim N(0, \sigma^2)$$

$$p_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \, \mathrm{e}^{\left(\frac{-x^2}{2\sigma^2}\right)} \tag{6.1}$$

6.1.1 Covariance Propagation

For preliminary filter formulation, given the complexity of determining the Jacobian of modified SGP4, the dynamic model used for covariance propagation is simplified by using a two-body formulation. Future work will develop the analytic Jacobian of modified SGP4, however, for initial filter design, a simplified model is used. Development of the two-body equation requires the following assumptions: 1) the mass of the satellite is negligible in comparison to the earth, 2) the coordinate system for the application is inertial, 3) the bodies of the satellite and earth are spherically symmetrical with uniform density, and 4) no other force acts on the system except for the gravitational force that acts along the line connecting the centers of the two-bodies. All two-body derivation is performed in GCRF.

Bate, Mueller, and White in [58] derive the two-body equation as:

$$\mathbf{r} = [r_1, r_2, r_3, \dot{r}_1, \dot{r}_2, \dot{r}_3]^T$$

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{6.2}$$

Equation 6.2 is a second order, ordinary differential equation (ODE) with three scalar components that can be expressed in Cartesian coordinates as:

$$\begin{aligned}
\ddot{r}_1 &= \frac{-\mu r_1}{(r_1^2 + r_2^2 + r_3^2)^{\frac{3}{2}}} \\
\ddot{r}_2 &= \frac{-\mu r_2}{(r_1^2 + r_2^2 + r_3^2)^{\frac{3}{2}}} \\
\ddot{r}_3 &= \frac{-\mu r_3}{(r_1^2 + r_2^2 + r_3^2)^{\frac{3}{2}}}
\end{aligned} \tag{6.3}$$

The equations of motion, 6.3, represent a set of three coupled, nonlinear, second order ODEs. In state space the functions are represented as:

$$\begin{aligned}
\mathbf{x} &= \mathbf{r} \\
\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), t)
\end{aligned} \tag{6.4}$$

The Jacobian of the nonlinear process function, $\mathbf{F}$, where r is the magnitude of the position vector, can then be calculated as:

$$\mathbf{F}\big|_{\mathbf{x}(t)} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}(t)}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\mu(2r_1^2 - r_2^2 - r_3^2)}{r^5} & \frac{3\mu r_1 r_2}{r^5} & \frac{3\mu r_1 r_3}{r^5} & 0 & 0 & 0 \\ \frac{3\mu r_1 r_2}{r^5} & \frac{\mu(-r_1^2 + 2r_2^2 - r_3^2)}{r^5} & \frac{3\mu r_2 r_3}{r^5} & 0 & 0 & 0 \\ \frac{3\mu r_1 r_3}{r^5} & \frac{3\mu r_2 r_3}{r^5} & \frac{\mu(-r_1^2 - r_2^2 + 2r_3^2)}{r^5} & 0 & 0 & 0 \end{bmatrix} \tag{6.5}$$

The filter derivation, presented in the next section, results in a Riccati Differential Equation (RDE) for covariance. The MATLAB function ode45.m, which uses the fourth order Runge-Kutta algorithm, presented in [59], will be used to solve the RDE to propagate the covariance. The Runge-Kutta algorithm will discretize and approximate the solution to the RDE.

6.2 Extended Kalman Filter

In [60], Gelb proves that the Kalman Filter (KF) produces the optimal state estimation, defined as minimizing the mean square estimation error, for linear systems with white, zero mean, and Gaussian process and measurement noise. However, most systems encountered in practice, including this tracking application, are nonlinear. The Extended Kalman Filter (EKF) was originally proposed by Stanley Schmidt and has become the most widely used nonlinear state estimation technique [61]. The basis of the EKF is that the dynamic and measurement model are linearized around the filter state update. There are filters that reduce the linearization errors of the EKF, however, at a cost of increased complexity and computation time. Based on the real-time nature of this application, the EKF was chosen. A brief summary of the EKF developed for this tracking application is presented below.

For an extensive derivation of the statistics involved in the EKF derivation consult work by Terejanu [62].

The system equations with continuous dynamics and discrete measurements are given as follows:

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{p}(t), t) + \mathbf{w} \\
\mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{q}_k) + \mathbf{v}_k \\
\mathbf{w} &\sim N(0, \mathbf{Q}) \\
\mathbf{v}_k &\sim N(0, \mathbf{R}_k)
\end{aligned} \tag{6.6}$$

where $\mathbf{p}$ and $\mathbf{q}$ each represent a set of deterministic, time-dependent parameters and the state vector, $\mathbf{x}$, is the position and velocity of the RSO.

6.2.1 Transformation of Variables

To begin the filter, the state and covariance must be initialized. The state is simply initialized as the SGP4 propagated state estimate at the beginning of the pass. The covariance, however, must be carefully considered. The most accurate initial covariance will produce the shortest convergence time, which is especially important for the short-arc passes being considered. The Cartesian GCRF does not provide an intuitive frame for a priori covariance estimation.

As shown in [38], the near in-track error dominates the position and velocity uncertainty, especially for LEO spacecraft. To account for this, the uncertainty must be formulated in a frame that allows for accurate modeling of the instantaneous in-track error. The spherical PQW frame, where $\hat{p}$ is aligned with the instantaneous orbital radius, $\hat{w}$ is aligned with the

orbit normal, and $\hat{q}$ is defined as the cross product of $\hat{w}$ and $\hat{p}$, is an ideal frame for this formulation. The covariance in the spherical PQW frame must be transformed to Cartesian GCRF to be used in subsequent filter derivation. Similar transformations using an orbit rotating Cartesian frame has been investigated [63], however, the spherical frame better represents the orbit path. The first step in the formulation of the covariance is a, time-dependent, transformation of spherical variables, $\mathbf{x}^S(t)$, to Cartesian variables, $\mathbf{x}^C(t)$. Note that time-dependent notation is dropped for conciseness. This can be calculated as:

$$\mathbf{x}^S = [r, \alpha, \delta, \dot{r}, \dot{\alpha}, \dot{\delta}]^T$$

$$\mathbf{x}^C = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$$

$$\mathbf{x}^S = \mathbf{g}(\mathbf{x}^C)$$

$$\begin{bmatrix} r \\ \theta \\ \phi \\ \dot{r} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \tan^{-1}\left(\dfrac{y}{x}\right) \\ \sin^{-1}\left(\dfrac{z}{\sqrt{x^2+y^2+z^2}}\right) \\ \dfrac{x\dot{x}+y\dot{y}+z\dot{z}}{\sqrt{x^2+y^2+z^2}} \\ \dfrac{\dot{x}y-x\dot{y}}{x^2+y^2} \\ \dfrac{\dot{z}(x^2+y^2)-z(x\dot{x}+y\dot{y})}{(x^2+y^2+z^2)\sqrt{x^2+y^2}} \end{bmatrix} \tag{6.7}$$

A full nonlinear pdf transformation has proven numerically intractable. Additionally, if the spherical state error is modeled as Gaussian, it will remain Gaussian only through a linear

transformation; Gaussian noise is an assumption made during the EKF formulation. As a result, a linearization of the transformation is calculated. The nonlinear transformation function can be linearized as follows:

$$\mathbf{G}\big|_{\mathbf{x}^C} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}^C}\right|_{\mathbf{x}^C}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 \\ \mathbf{G}_3 & \mathbf{G}_4 \end{bmatrix}$$

$$\mathbf{G}_1 = \begin{bmatrix} \dfrac{x}{\sqrt{x^2+y^2+z^2}} & \dfrac{y}{\sqrt{x^2+y^2+z^2}} & \dfrac{z}{\sqrt{x^2+y^2+z^2}} \\ \dfrac{-y}{x^2+y^2} & \dfrac{x}{x^2+y^2} & 0 \\ \dfrac{-xz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} & \dfrac{-yz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} & \dfrac{\sqrt{x^2+y^2}}{x^2+y^2+z^2} \end{bmatrix}$$

$$\mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_3 = \begin{bmatrix} \dfrac{\dot{x}y^2 - x\dot{y}y + \dot{x}z^2 - x\dot{z}z}{\sqrt[3]{x^2+y^2+z^2}} & \dfrac{\dot{y}x^2 - \dot{x}yx + \dot{y}z^2 - y\dot{z}z}{\sqrt[3]{x^2+y^2+z^2}} & \dfrac{zx^2 - \dot{x}zx + \dot{z}y^2 - \dot{y}zy}{\sqrt[3]{x^2+y^2+z^2}} \\ -\dfrac{-\dot{y}x^2 + 2\dot{x}xy + \dot{y}y^2}{(x^2+y^2)^2} & \dfrac{\dot{x}x^2 + 2\dot{y}xy - \dot{x}y^2}{(x^2+y^2)^2} & 0 \\ \mathbf{G}_3(3,1) & \mathbf{G}_3(3,2) & \mathbf{G}_3(3,3) \end{bmatrix}$$

$$\mathbf{G}_3(3,1) = -\frac{-\dot{z}x^5+2\dot{x}x^4z-2\dot{z}x^3y^2+3\dot{y}x^3yz+\dot{z}x^3z^2+\dot{x}x^2y^2z-\dot{z}xy^4+3\dot{y}xy^3z+\dot{z}xy^2z^2+\dot{y}xyz^3-\dot{x}y^4z-\dot{x}y^2z^3}{\sqrt[3]{x^2+y^2}(x^2+y^2+z^2)^2}$$

$$\mathbf{G}_3(3,2) = -\frac{-\dot{z}x^4y-\dot{y}x^4z+3\dot{x}x^3yz-2\dot{z}x^2y^3+\dot{y}x^2y^2z+\dot{z}x^2yz^2-\dot{y}x^2z^3+3\dot{x}xy^3z+\dot{x}xyz^3-\dot{z}y^5+2\dot{y}y^4z+\dot{z}y^3z^2}{\sqrt[3]{x^2+y^2}(x^2+y^2+z^2)^2}$$

$$\mathbf{G}_3(3,3) = \frac{\dot{x}x^3+\dot{y}x^2y+2\dot{z}x^2z+\dot{x}xy^2-\dot{x}xz^2+\dot{y}y^3+2\dot{z}y^2z-\dot{y}yz^2}{\sqrt{x^2+y^2}(x^2+y^2+z^2)^2}$$

$$\mathbf{G}_4 = \begin{bmatrix} \dfrac{x}{\sqrt{x^2+y^2+z^2}} & \dfrac{y}{\sqrt{x^2+y^2+z^2}} & \dfrac{z}{\sqrt{x^2+y^2+z^2}} \\ \dfrac{y}{x^2+y^2} & \dfrac{-x}{x^2+y^2} & 0 \\ \dfrac{xz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} & \dfrac{yz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} & -\dfrac{\sqrt{x^2+y^2}}{x^2+y^2+z^2} \end{bmatrix} \tag{6.8}$$

The linearized map from spherical to Cartesian can be used to transform the covariance as:

$$\mathbf{P}^{C,PQW} = \mathbf{G}\big|_{\mathbf{x}^C}^{T} \mathbf{P}^{S,PQW} \mathbf{G}\big|_{\mathbf{x}^C} \tag{6.9}$$

Once the covariance is transformed into Cartesian PQW it can then be rotated to the GCRF, based on the, time-dependent, eccentricity and angular momentum unit vectors, as follows:

$$\mathbf{R}_{PQW}^{GCRF} = \left[\hat{\mathbf{e}}^{GCRF} \mid \hat{\mathbf{h}}^{GCRF} \times \hat{\mathbf{e}}^{GCRF} \mid \hat{\mathbf{h}}^{GCRF}\right]$$

$$\mathbf{P}^{GCRF} = \mathbf{R}_{PQW}^{GCRF} \mathbf{P}^{C,PQW} (\mathbf{R}_{PQW}^{GCRF})^T \tag{6.10}$$

6.2.2 Propagation and Measurement Update

After initialization, for each timestep $k = 1, 2, ...$, the state estimate and covariance are propagated from time t_{k-1}^{+} to t_k^{-}, where the superscript denotes whether the estimate is *a*

priori or *a posteriori*, with respect to inclusion of the measurement at timestep k, $\mathbf{y}_k$. The process and covariance are propagated using:

$$\begin{aligned} \mathbf{x}_k^- &= \mathbf{SGP4}^*(\mathbf{x}_{k-1}^+, t) \\ \dot{\mathbf{P}} &= \mathbf{F}\big|_{\mathbf{x}_{k-1}^+} \mathbf{P}_{k-1}^+ + \mathbf{P}_{k-1}^+ \mathbf{F}\big|_{\mathbf{x}_{k-1}^+}^T + \mathbf{Q} \end{aligned} \tag{6.11}$$

Then the measurement is assimilated into the estimate as follows:

$$\begin{aligned} \mathbf{x}_k^+ &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k^-)) \\ \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}\big|_{\mathbf{x}_k^-}^T [\mathbf{H}\big|_{\mathbf{x}_k^-} \mathbf{P}_k^- \mathbf{H}\big|_{\mathbf{x}_k^-}^T + \mathbf{R}_k]^{-1} \\ \mathbf{P}_k^+ &= [\mathbf{I} - \mathbf{K}_k \mathbf{H}\big|_{\mathbf{x}_k^-}] \mathbf{P}_k^- \end{aligned} \tag{6.12}$$

where $\mathbf{K}_k$ is called the Kalman Gain and $\mathbf{H}$ is the Jacobian of the nonlinear measurement model function. For simulation purposes this is defined as the GCRF to TCRF transformation, presented in section 4.2.1, with time dependence dropped for conciseness. However, for implementation the full measurement model from GCRF state to image plane state, presented in Chapter 5, must be used. The Jacobian, where all vectors are in GCRF, can be calculated as:

$$\mathbf{H}\big|_{\mathbf{x}_k} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}_k}$$

$$\boldsymbol{\rho} = \mathbf{r}_{sat} - \mathbf{r}_{site}$$

$$\dot{\boldsymbol{\rho}} = \mathbf{v}_{sat} - \mathbf{v}_{site}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix}$$

$$\mathbf{H}_1 = \begin{bmatrix} \dfrac{\rho_1\rho_2}{\sqrt{\frac{\rho_1^2}{\rho_1^2+\rho_2^2}}(\rho_1^2+\rho_2^2)^{\frac{3}{2}}} & \dfrac{\sqrt{\frac{\rho_1^2}{\rho_1^2+\rho_2^2}}}{\sqrt{\rho_1^2+\rho_2^2}} & 0 \\ -\dfrac{\rho_1\rho_3}{\rho^3} & -\dfrac{\rho_2\rho_3}{\rho^3} & \dfrac{\rho_1^2\rho_2^2}{\rho^3} \\ \dfrac{2\dot{\rho}_1\rho_1\rho_2+\dot{\rho}_2(-\rho_1^2+\rho_2^2)}{(\rho_1^2+\rho_2^2)^2} & \dfrac{2\dot{\rho}_2\rho_1\rho_2+\dot{\rho}_1(-\rho_1^2+\rho_2^2)}{(\rho_1^2+\rho_2^2)^2} & 0 \\ \mathbf{H}_1(4,1) & \mathbf{H}_1(4,2) & \mathbf{H}_1(4,3) \end{bmatrix}$$

$$\mathbf{H}_1(4,1) = \frac{\rho_1\left[-\dot{\rho}_3\rho^3+\rho_3\left((\rho_1^2+\rho_2^2)\rho\cos\left(\frac{\rho_3}{\rho}\right)+\rho_3\rho^2\sin\left(\frac{\rho_3}{\rho}\right)\right)\right]}{(\rho_1^2+\rho_2^2)^{\frac{3}{2}}\rho^3}$$

$$\mathbf{H}_1(4,2) = \frac{\rho_2\left[-\dot{\rho}_3\rho^3+\rho_3\left((\rho_1^2+\rho_2^2)\rho\cos\left(\frac{\rho_3}{\rho}\right)+\rho_3\rho^2\sin\left(\frac{\rho_3}{\rho}\right)\right)\right]}{(\rho_1^2+\rho_2^2)^{\frac{3}{2}}\rho^3}$$

$$\mathbf{H}_1(4,3) = -\frac{(\rho-\rho_3)(\rho+\rho_3)\cos\left(\frac{\rho_3}{\rho}\right)+\rho\rho_3\sin\left(\frac{\rho_3}{\rho}\right)}{\rho^2\sqrt{(\rho-\rho_3)(\rho+\rho_3)}}$$

$$\mathbf{H}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\dfrac{\rho_2}{\rho_1^2+\rho_2^2} & \dfrac{\rho_1}{\rho_1^2+\rho_2^2} & 0 \\ 0 & 0 & \dfrac{1}{\sqrt{\rho_1^2+\rho_2^2}} \end{bmatrix} \tag{6.13}$$

6.3 Simulation

The application of the derived EKF was simulated in MATLAB. Realistic pass parameters and noisy measurements were determined using the developed observable passes MATLAB function. An observable pass over two and a half minutes is considered with measurements every six seconds and a TLE epoch approximately two hours before the pass. Measurement uncertainty was based on an analysis of sensor uncertainty, presented by Vallado [30]. EKF simulation results are presented in Figures 6.1 to 6.12. Initial results suggest that accuracy is increased and uncertainty is decreased over the pass, however, this is not true for the third velocity component and filter convergence is sensitive to uncertainty estimates. It is possible that the short time span of the considered pass does not allow the filter to sufficiently converge. Additionally, modeling the uncertainty of SGP4 is difficult and further analysis must be performed to assess the possibility of reducing uncertainty and increasing tracking accuracy by implementing statistical OD in real time.

In summary, this chapter has presented the background information and derivations needed to simulate and apply an EKF for real-time OD implementation. Contributions of this chapter include a novel application of sequential SGP4, spherical covariance transformation, and simulation of realistic RSO pass data for the Blacksburg Observatory. Preliminary results show that real-time OD could increase pointing accuracy, however, the simulation has shown the results are sensitive to measurement and SGP4 uncertainty estimates. The techniques presented in this chapter can also be used to create off-line OD over multiple passes.

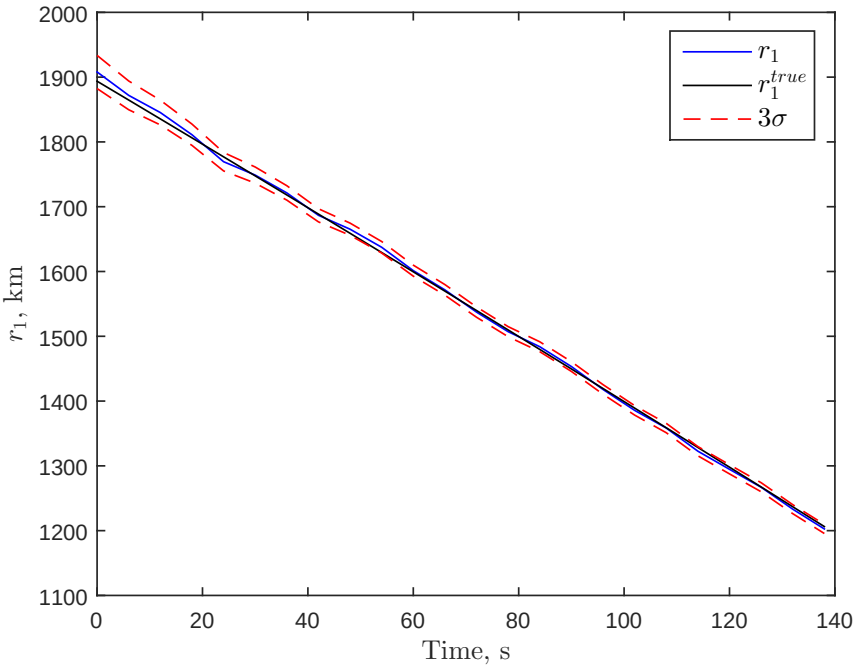

Figure 6.1: Estimated and True r_1 vs. Time

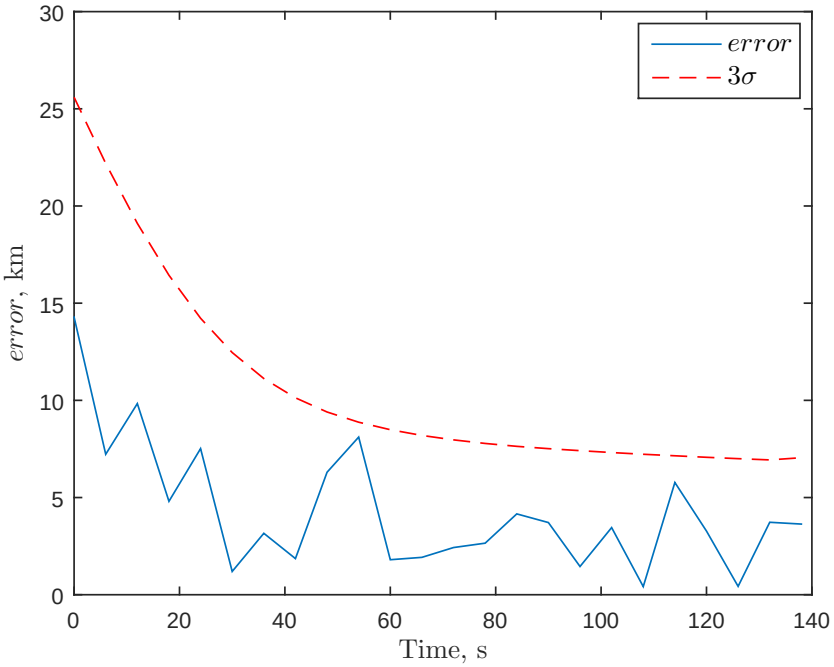

Figure 6.2: Estimated r_1 Absolute True Error vs. Time

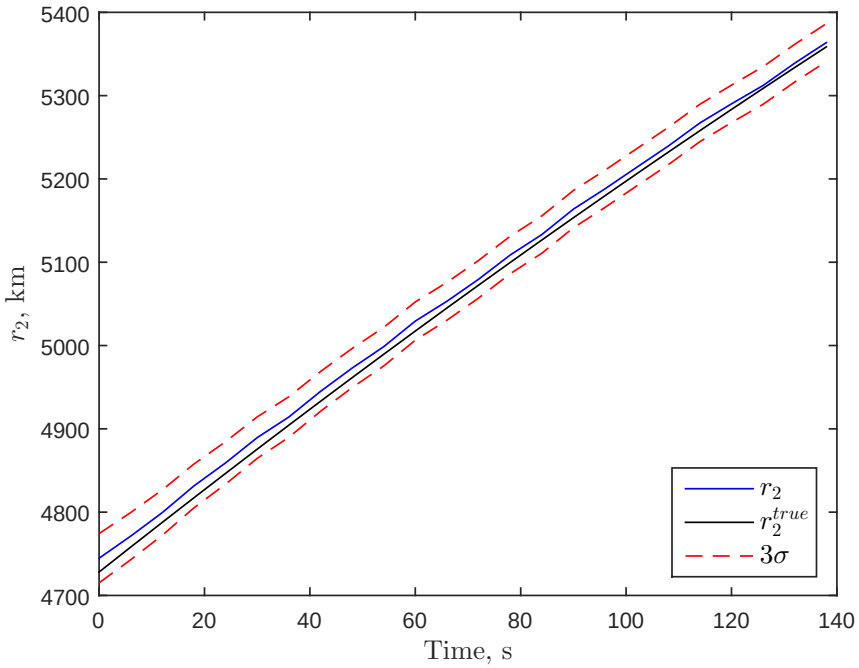

Figure 6.3: Estimated and True r_2 vs. Time

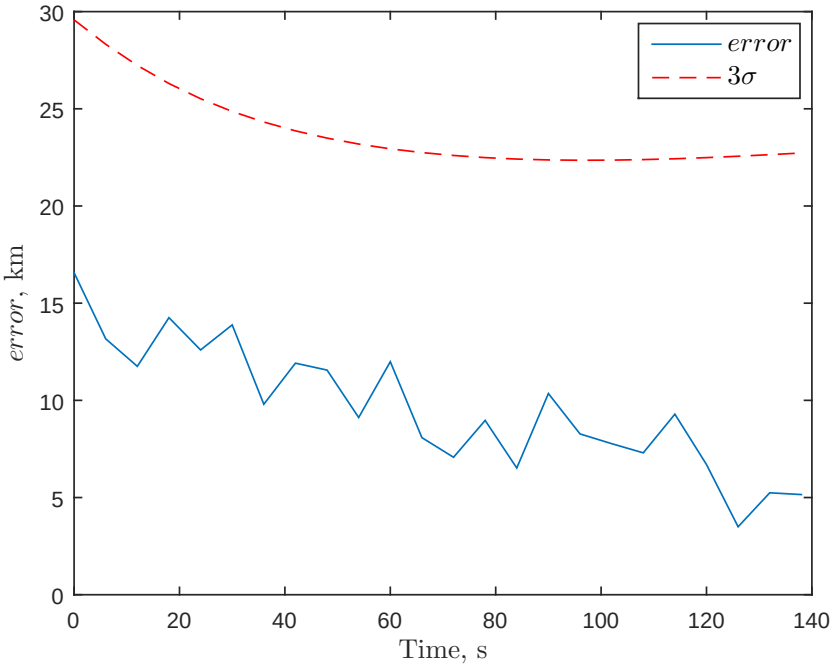

Figure 6.4: Estimated r_2 Absolute True Error vs. Time

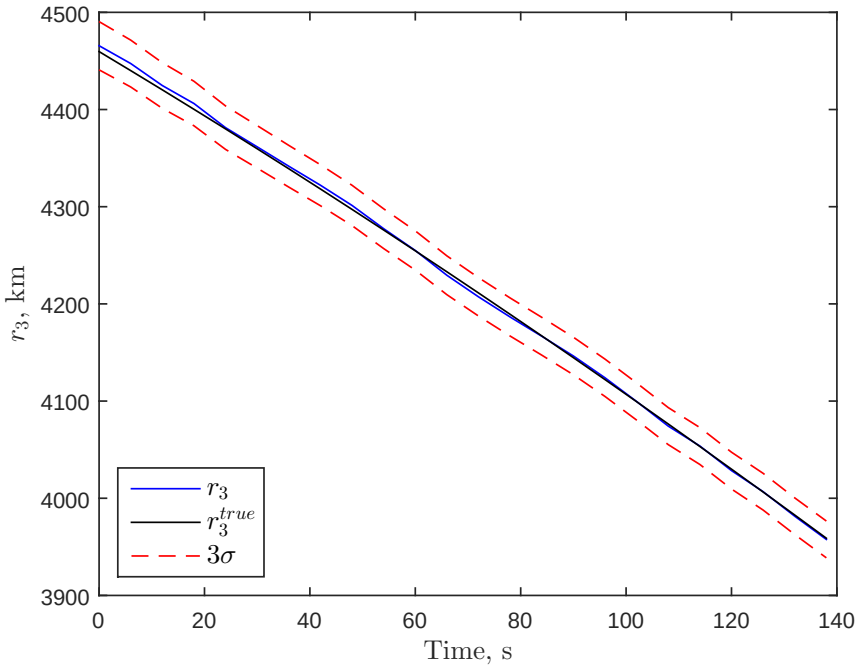

Figure 6.5: Estimated and True r_3 vs. Time

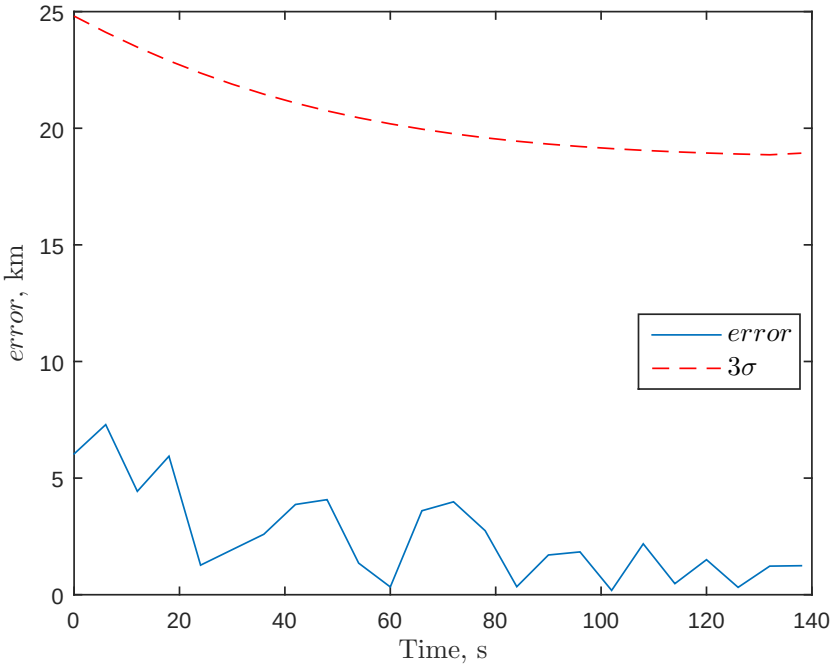

Figure 6.6: Estimated r_3 Absolute True Error vs. Time

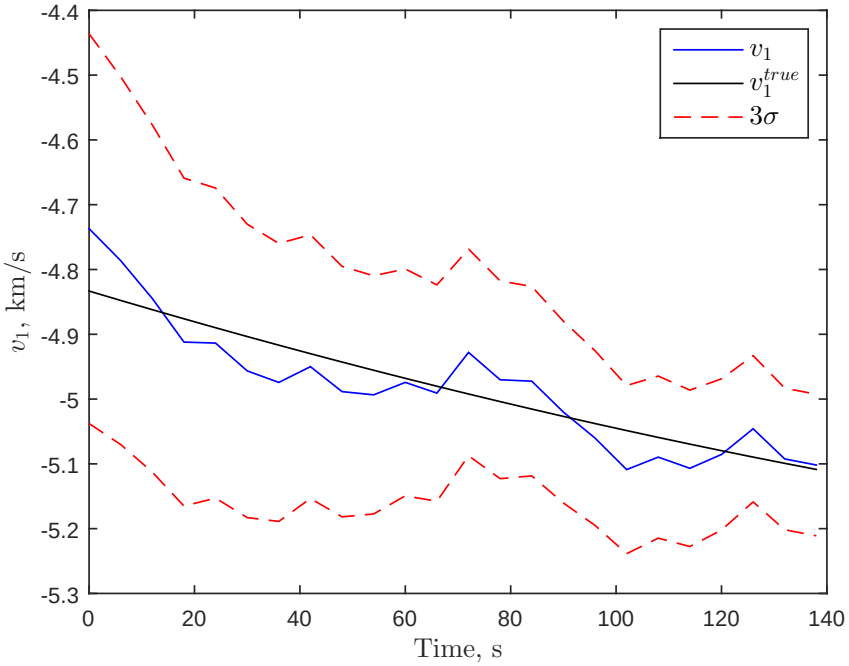

Figure 6.7: Estimated and True v_1 vs. Time

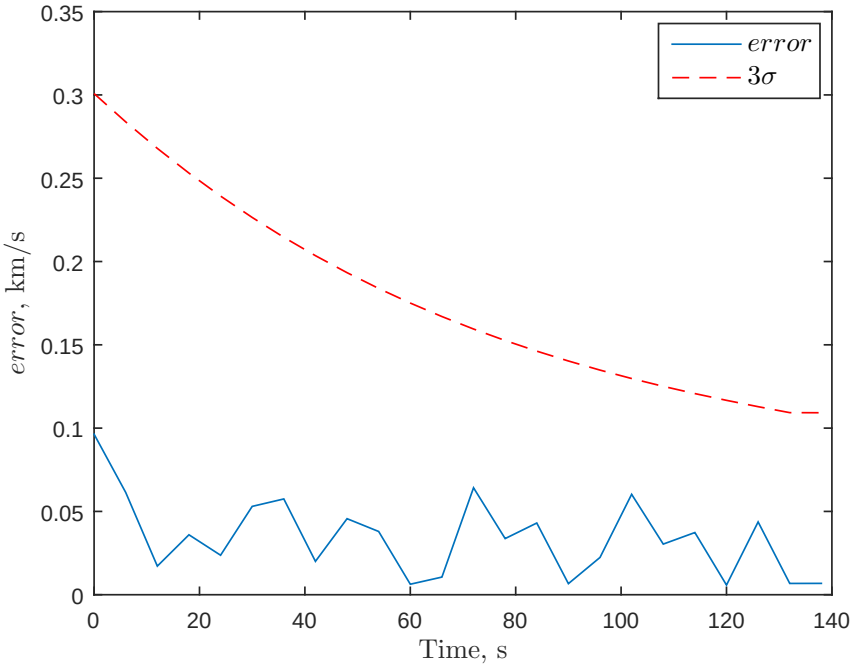

Figure 6.8: Estimated v_1 Absolute True Error vs. Time

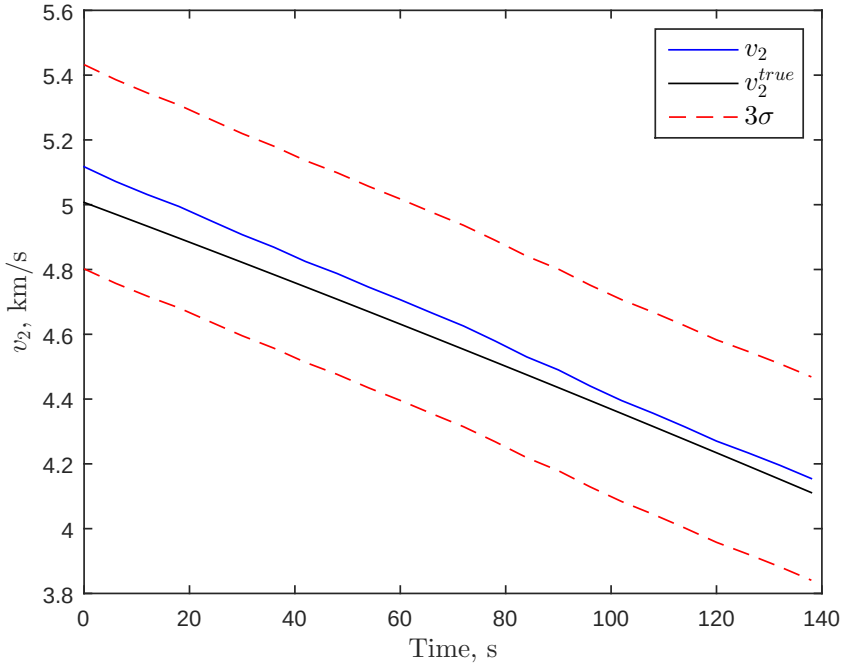

Figure 6.9: Estimated and True v_2 vs. Time

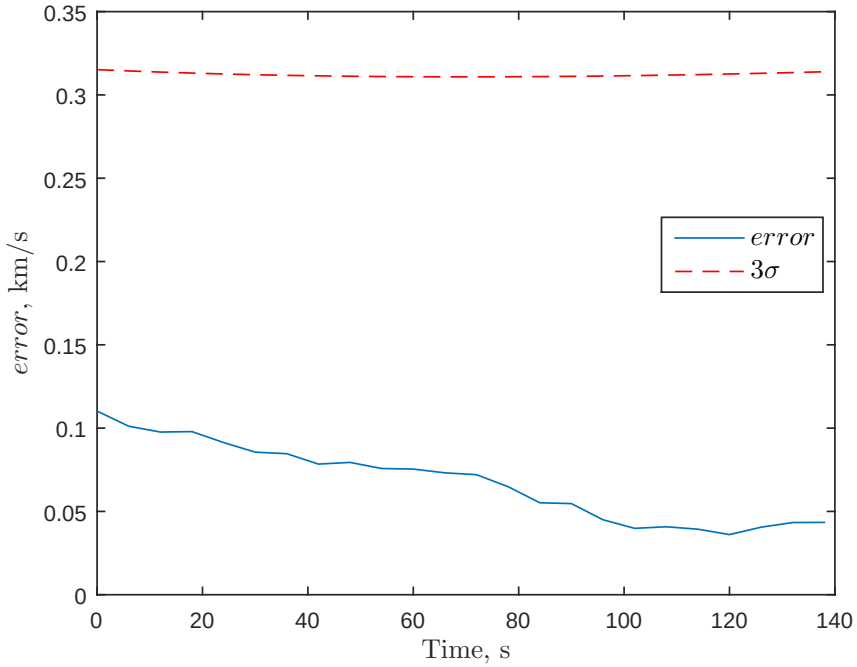

Figure 6.10: Estimated v_2 Absolute True Error vs. Time

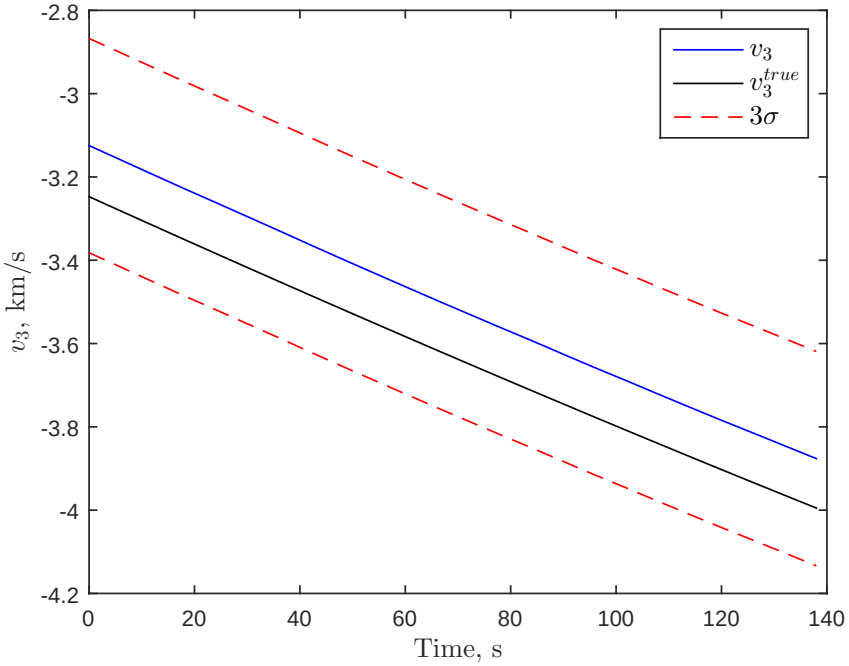

Figure 6.11: Estimated and True v_3 vs. Time

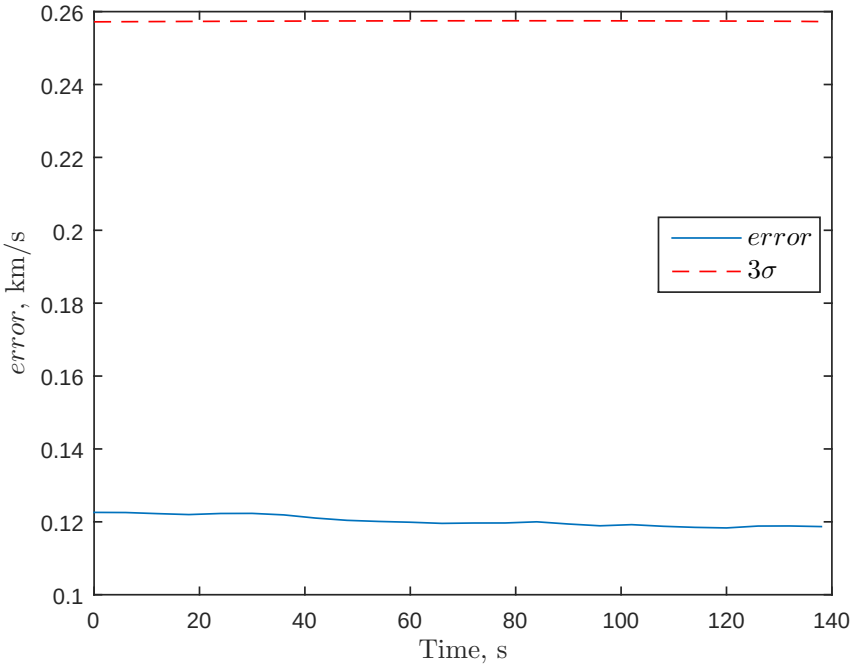

Figure 6.12: Estimated v_3 Absolute True Error vs. Time

Chapter 7

Conclusions

7.1 Summary

This thesis has presented a system design overview of the Virginia Tech Optical Satellite Tracking Telescope. Research was performed on previous COTS based telescope projects, including Raven-class telescopes, as an initial starting point for VTOST hardware design. Specific hardware was then selected based on design parameters derived from the requirements of VTOST, which include quality satellite image data, autonomous operation, and ease of use.

The literature review, background research, and algorithms required for surveillance operations were presented. These algorithms included determining RSO position from orbital propagation; transforming RSO states from GCRF to ITRF and SEZ to determine viewing geometry; and Sun GCRF position analysis to determine RSO observability windows. Additionally, all necessary MATLAB code was developed as an accompaniment to this thesis and is presented along with full operational procedures for VTOST surveillance data collection.

Surveillance operation was verified and data was collected for a number of RSOs. The data collected thus far, exemplified by Figure 4.2, has validated the VTOST system design and has demonstrated the feasibility of future research on expanding VTOST capabilities.

A closed-loop tracking control solution for the current hardware setup with limited velocity control and the groundwork for tracking implementation with full velocity control has been presented. Additionally, the background information and derivations needed to simulate and apply an EKF for real-time OD implementation, for this application, has been presented. Contributions of this thesis include a novel application of sequential SGP4, spherical covariance transformation, and simulation of realistic RSO pass data for the Blacksburg Observatory. Preliminary results show that real-time OD could increase pointing accuracy, however, the simulation has shown the results are sensitive to measurement and SGP4 uncertainty estimates. The techniques presented can also be used to create off-line OD over multiple passes.

In sum, this thesis has presented the theoretical background, preliminary algorithms, and initial control design needed for a full tracking implementation. Additionally, preliminary considerations are presented for resolved imagery and autonomous, remote operation. Overall, this thesis has 1) demonstrated the feasibility of data collection from COTS hardware, 2) presented working surveillance operational code ready for further data collection, and 3) outlined the further steps needed to institute full tracking implementation and remote, autonomous tasking.

7.2 Future Work

A number of future work steps remain to institute full tracking implementation and remote, automatic tasking. To enable fully automatic tasking the observatory must be connected

to the AOE network. Currently, there is no network at the observatory, and thus no way to remotely connect from a ground station to a control laptop located at the observatory. Additionally, the observatory dome and dome slit control must be computerized and coupled with mount pointing. Preliminary design considerations for this work step have been presented in this paper.

For full tracking implementation, the VTOST hardware and control software described in this thesis and the image processing platform, GEODETICA, presented in Sease's dissertation, [51], must be analyzed and synthesized. Researchers can access the ActiveX objects, developed in this thesis, for control options. Once velocity control has been developed, researchers can implement the pointing update tracking algorithms presented in this thesis. It is estimated that developing velocity control and implementing GEODETICA will take a team of two to three dedicated researchers approximately 4 months to complete. In this thesis, data has been collected for use by future researchers to assist in understanding and implementing GEODETICA for VTOST tracking. Once tracking has been implemented, statistical orbit determination may be able to be used to increase tracking accuracy.

Finally, to allow the acquisition of resolved imagery, a camera for attachment to the back of the telescope and the Celestron Deluxe Tele-Extender should be purchased and camera control software developed. Once tracking is implemented, it is estimated that developing camera control software, calibrating image parameters, and collecting resolved imagery will take a team of two to three dedicated researchers approximately 4 months to complete. For VTOST, the recommended camera for purchase is the Celestron NexImage 5 Solar System Imager Camera and a Celestron X-Cel LX 1.25" 3x Barlow Lens. This camera will provide a small FOV and one of the lowest meter per pixel values in the $200 to $300 camera range. This thesis presents initial considerations for resolved imagery, including FOV analysis and ISO and exposure settings.

References

[1] T. Blake, "Orbit Outlook," *DARPA Tactical Technology Office*, 2014.

[2] D. Sibert et al., "Collaborative Commercial Space Situational Awareness with ESpOC-Empowered Telescopes," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2013.

[3] Joint Chiefs of Staff, "Space Operations," *Joint Publication 3-14*, 2009.

[4] B. Obama, "National Space Policy of the United States of America," *Executive Office the the President*, 2010.

[5] S. Kan, "China's Anti-Satellite Weapon Test," *Congressional Research Service, Report for Congress*, April 2007.

[6] T. S. Kelso, "Analysis of the Iridium 33-Cosmos 2251 Collision," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2009.

[7] USSTRATCOM, "Space Control and Space Surveillance," *Report*, 2014.

[8] M. Clifford, D. Baiocchi, and W. Welser, "A Sixty-Year Timeline of the Air Force Maui Optical and Supercomputing Site," *Rand Corporation Report*, 2013.

[9] R. D. Coder and M. J. Holzinger, "Sizing of a Raven-class Telescope Using Performance Sensitivities," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2013.

[10] D. Nishimoto, D. Archambeault, D. Gerwe, and P. Kervin, "Satellite Attitude from a Raven Class Telescope," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2010.

[11] J. Africano et al., "AMOS Debris Observation," *Third European Conference on Space Debris*, March 2001.

[12] M. L. Thrall, "Orbit Determination of Highly Eccentric Orbits using a RAVEN Telescope," *Naval Postgraduate School, Master's Thesis*, September 2005.

[13] C. T. Bellows, "Leveraging External Sensor Data for Enhanced Space Situational Awareness," *Air Force Institute of Technology, Doctorate Dissertation*, September 2015.

[14] S. Bruski et al., "An Optical Satellite Tracking System for Undergraduate Research," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2012.

[15] Sensors and Imaging Group, C. S. Draper Laboratory, *Personal Correspondence.*

[16] The Imaging Source, *DMK 41BU02 Datasheet.*

[17] Sony Corporation, *ICX205AL Datasheet.*

[18] J. R. Wertz, *Orbit and Constellation Design and Management.* New York, NY: Springer, 2001.

[19] National Aeronautics and Space Administration, *About the Space Station: Facts and Figures.*

[20] D. Lang et al., "Astrometry.net: Blind Astrometric Calibration of Arbitrary Astronomical Images," *The Astronomical Journal, Vol. 139*, March 2010.

[21] Celestron, *EdgeHD Series Instruction Manual*.

[22] Celestron, *CGE Pro Series Instruction Manual*.

[23] D. R. Williams, *Jupiter Fact Sheet*. National Space Science Data Center.

[24] D. Campbell, *Field of View Calculator*. 12 Dimensional String – Amateur Astronomy.

[25] D. D. McCarthy and G. Petit, *IERS Conventions (2003)*. International Earth Rotation Service Technical Note 32, 2004.

[26] M. Feissel and F. Mignard, "The Adoption of ICRS on 1, January 1998: Meaning and Consequences," *Astronomy and Astrophysics*, vol. 331, 1997.

[27] P. K. Seidelmann and J. Kovalecsky, "Application of the New Concepts and Definitions (ICRS, CIP, and CEO) in Fundamental Astronomy," *Astronomy and Astrophysics*, vol. 332, 2002.

[28] J. R. Wertz and W. J. Larson, *Space Mission Analysis and Design. 3rd ed.* New York, NY: Springer, 2010.

[29] K. M. Borkowski, "Accurate Algorithms to Transform Geocentric to Geodetic Coordinates," *Journal of Geodesy*, March 1989.

[30] D. A. Vallado, *Fundamentals of Astrodynamics and Applications, 4th ed.* Hawthorne, CA: Microcosm Press.

[31] The U.S. Naval Observatory, *The Astronomical Almanac*. U.S. Navy, 2015.

[32] W. E. Wiesel, *Spaceflight Dynamics, 3rd ed.* Beavercreek, OH: Aphelion Press, 2010.

[33] C. Peat, *heavens-above.com*.

[34] F. R. Hoots, P. W. J. Schumacher, and R. A. Glover, "History of Analytical Orbit Modeling in the U.S. Space Surveillance System," *Journal of Guidance, Control, and Dynamics Vol. 27*, March 2004.

[35] D. Brouwer, "Solution of the Problem of Artificial Satellite Theory Without Drag," *Astronomical Journal Vol. 64*, October 1959.

[36] Y. Kozai, "The Motion of a Close Earth Satellite," *Astronomical Journal Vol. 64*, October 1959.

[37] F. R. Hoots and R. L. Roehrich, *Spacetrack Report No. 3 – Models for Propagation of NORAD Element Sets*. North American Aerospace Defense Command, 1988.

[38] D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso, "Revisiting Spacetrack Report 3," *AIAA/AAS Astrodynamics Specialist Conference*, August 2006.

[39] D. A. Vallado and C. Paul, "SGP4 Orbit Determination," *AIAA/AAS Astrodynamics Specialist Conference*, August 2008.

[40] T. S. Keslo, "Validation of SGP4 and IS-GPS-200D Against GPS Precision Ephemerides," *AIAA/AAS Space Flight Mechanics Conference*, January 2007.

[41] M. R. Greene and R. E. Zee, "Increasing the Accuracy of Orbital Position Information from NORAD SGP4 Using Intermittent GPS Readings," *AIAA/USU Conference on Small Satellites*, 2009.

[42] D. Oltrogge and J. Ramrath, "Parametric Characterization of SGP4 Theory and TLE Positional Accuracy," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2014.

[43] S. Herrick, *Astrodynamics: Orbit Correction, Perturbation Theory, Integration Vol. II.* London, UK: Van Nostrand Reinhold Co., 1972.

[44] J. H. Seago and D. A. Vallado, "Coordinate Frames of the U.S. Space Object Catalog," *Astrodynamics Specialist Conference and Guidance, Navigation, and Control Conference*, 2000.

[45] A. C. Holland, "The Effects of Atmospheric Refraction on Angles Measured from a Satellite," *Journal of Geophysical Research*, December 1961.

[46] C. L. Doolittle, "The Constant of Aberration," *Astronomical Journal Vol. 24*, March 1905.

[47] K. Lang, *Astrophysical Formulae: Space, Time, Matter and Cosmology, 3rd ed.* New York, NY: Springer, 2013.

[48] B. Wallace, R. Scott, and A. Spaans, "Automatic Recognition of Low Earth Orbit Objects from Image Sequences," *IEEE International Conference on Intelligent Computer Communication and Processing*, August 2011.

[49] B. Wallace, R. Scott, and A. Spaans, "The DRDC Ottawa Space Surveillance Observatory," *Advanced Maui Optical and Space Surveillance Technologies Conference*, September 2007.

[50] B. Sease and B. Flewelling, "GEODETICA: A General Software Platform for Processing Continuous Space-Based Imagery," *25th AAS/AIAA Space Flight Mechanics Meeting*, January 2015.

[51] B. Sease, *Doctorate Dissertation.* Blacksburg, VA: Virginia Tech.

[52] D. W. Hogg et al., "Automated Astronomy," *Astronomical Data Analysis Software and Systems ASP Conference Series, Vol. 394*, September 2007.

[53] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems, 2nd ed.* Boca Raton, FL: CRC Press, 2012.

[54] P. Singla, D. T. Griffith, J. L. Crassidis, and J. J. L., "Attitude Determination and Autonomous On-Orbit Calibration of Star Tracker for GIFTS Mission," *AIAA/AAS Space Flight Mechanics Meeting*, January 2002.

[55] B. Sease, R. Koglin, and B. Flewelling, "Long Integration Star Tracker Image Processing for Combined Attitude - Attitude Rate Estimation," *Sensors and Systems for Space Applications IV, SPIE Vol. 8739*, April 2013.

[56] R. Berry and J. Burnell, *The Handbook of Astronomical Image Processing.* Richmond, VA: Willman-Bell, Inc.

[57] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and Systems.* London, UK: Prentice-Hall.

[58] R. R. Bate, D. D. Mueller, and J. E. White, *Fundamentals of Astrodynamics.* New York, NY: Dover Publications, 1971.

[59] S. C. Chapra, *Numerical Methods, 3rd ed.* New York, NY: McGraw-Hill Education, 2011.

[60] A. Gelb, J. Kasper Jr., R. A. Nash Jr., C. F. Price, and A. A. Sutherland Jr., *Applied Optimal Estimation.* Cambridge, MA: The M.I.T. Press.

[61] D. Simon, *Optimal State Estimation.* Hoboken, New Jersey: John Wiley and Sons, 2006.

[62] G. A. Terejanu, *Extended Kalman Filter Tutorial.* University at Buffalo.

[63] D. A. Vallado, "Covariance Transformations for Satellite Flight Dynamics Operations," *AAS/AIAA Astrodynamics Specialist Conference*, August 2003.

Appendix A

Installation Procedures

Internet links and current software versions are provided as of 4 April 2016, when the new VTOST control laptop was received and set up, and may have since changed. Information will be provided so that appropriate new links can be found. Installation procedure has been verified for 64 bit Windows 7 Enterprise with MATLAB R2015a and 64 bit Windows 10 Pro with MATLAB R2016a. The procedure below is presented for the currently designated VTOST control laptop (AOE domain name: bigdipper) with Windows 10 Pro 64 bit and MATLAB R2016a. Note that currently, MATLAB will need to be installed using the Virginia Tech individual license of the operator. This is because there are issues with using the network license when not connected to the internet. Once these procedures are complete, connect the camera and telescope through their respective methods. The MATLAB VTOST operation code will initialize both devices and work from there.

ASCOM Celestron Telescope Driver

Astronomy Common Object Model (ASCOM) is a group of developers and astronomical instrument makers that work together to create vendor and language independent standards for drivers that provide plug and play control of astronomical devices. For VTOST purposes, this means drivers are available to call and create a COM server in MATLAB for telescope control.

On the ASCOM main website, $http://ascom-standards.org/$, download the "ASCOM Platform 6.2" and perform a standard install.

In the ASCOM Downloads Telescope/Mount Drivers site, $http://ascom-standards.org/Downloads/ScopeDrivers.htm$, download the "Celestron Unified (6.0.5604)" driver and perform a standard install.

The first time you connect the mount to the control computer you have to initialize the driver through the COM port by running ASCOM.DriverConnect in "C:\Program Files (x86)\ASCOM\Platform 6\Tools textbackslashDriverConnect64". Note: you have to use the same USB port in which you initialized the driver. You have to reinitialize if you wish to switch.

The Imaging Source Software and Camera Driver

For the camera, the driver operate the cameara and the driver to call and create a COM server in MATLAB are desired. Additionally, The Imaging Source provides a software platform for camera control and image acquisition, "IC Imaging, Image Acquisition", that is useful for camera calibration and other purposes. All downloads from The Imaging Source can be found in their download center, $http://www.theimagingsource.com/en_US/support/downloads/$.

First from, *http* : *//www.theimagingsource.com/en_US/support/downloads/details/icwdmuvccamtis/*, download the "Device Driver for The Imaging Source USB Cameras" (current version 2.8.9). Extract files from zipped folder and perform a standard install by connecting the camera to the computer and running "drvInstaller.exe" from the main folder.

Next, to download the camera control software from, *http* : *//www.theimagingsource.com/en_US/support/downloads/details/iccapture/*, download the "IC Capture, Image Acquisition" (current version 2.4.633.2555) and perform a standard install.

Next, to download the ActiveX control from, *http* : *//www.theimagingsource.com/en_US/support/downloads/details/icimagingcontrolactivex/*, download the "IC Imaging Control ActiveX" (current version 3.2.0.0) and perform a standard install.

Next, to download the ActiveX control runtime setup from, *http* : *//www.theimaging source.com/en_US/support/downloads/details/icimagingcontrolactivexrt/*, download the "IC Imaging Control ActiveX Runtime Setup" (current version 3.2.0.0) and perform a standard install (allow install to same ActiveX folder, if asked).

Finally, to download the MATLAB plugin from, *http* : *//www.theimagingsource.com/en_US/support/downloads/details/icmatlabr*2013*b/*, download the "IC Matlab Plugin for Matlab R2013b and R2014" (current version 3.3.0.43) and perform a standard install.

USB to RS-232 Connection

The USB to RS-232 cable is made by Trendnet and the driver can be found at the download section of the TU-S9 (version v2.0R) download page, *http* : *//www.trendnet.com/support/supportdetail.asp?prod* = 265_*TU*−*S*9. Download the "Driver/Utility" zip folder. Unzip the downloaded folder and perform a standard install by running "TU-S9_PL2303_Prolific_Driver

Installer_v1.12.0.exe" from the "Windows" sub-folder.

GEODETICA and Astrometry.net

First, contact Dr. Black to make sure you have permission to utilize GEODETICA code (it is under AFRL distribution) and to obtain Brad Sease's "GEODETICA_Code" folder and a copy of the "indexfiles" folder, the star catalog fits files (∼30 GB). Place the "GEODETICA_Code" folder in the "C:\" directory.

From "GEODETICA_Code\AstNet Cygwin Installer\Source" run "setup-x86.exe". Select "Install from Internet". Install for all uses to "C:\cygwin". Select local package directory "GEODETICA_Code" and then use direct internet connection. Select $ftp://mirror.cs.vt.edu$ from the list. Next open "localsetup.bat" in Notepad and select all of the packages for Cygwin Setup listed at the top of that document (for the gcc package select the gcc4 package in the obsolete section).

After installation copy all ".gz", ".sh", and ".bz2" files from "GEODETICA_Code\AstNet Cygwin Installer\Source" to "C:\cygwin\home \build" (create this folder). Then place the fits "indexfiles" folder into "C:\cygwin\home". Then open command prompt and enter "cygwin\bin\bash –login -c '/home/build/setup.sh'". After the build you should be able to call the MATLAB functions that call Astrometry.net software.

Camera Mounting Setup

The camera is fastened onto an ADM Accessories Losmandy "D" Series Saddle using a 1/4"-20 x 1/2" socket head cap screw, which uses a 3/16" hex wrench, in the second from the

front bore on the camera and the front bore of the saddle. The ADM Saddle is then used to mount the camera onto the telescope via the dovetail bar on top of the telescope. Finally, the aperture pin is compressed using a 5/16" hex nut screw and a 5/16"-18 x 5/16" socket head set screw, which uses a 5/32" hex wrench.

Appendix B

MATLAB Function Listings

TEME to GCRF Coordinate Transformation

```
function [r_gcrf,v_gcrf] = sgp4_teme2gcrf(r_sgp4,v_sgp4,jdk,d_at,...
dX_interp,dY_interp,mjd_interp,iau_data,reverse)

% D. Luciani

% inputs:
% jdk, julian date of utc time for timestep
% r_sgp4, 1x3 teme position vector
% v_sgp4, 1x3 teme velocity vector
% a_sgp4, 1x3 teme acceleration vector
% d_at, TAI-UTC difference in seconds (from NGA init)
% dX_interp, Bull. A dX wrt IAU2000A Nutation (msec. of arc),
%       Free Core Nutation NOT Removed
% dY_interp, Bull. A dY wrt IAU2000A Nutation (msec. of arc)
% mjd_interp, modified julian days of update epoch for day
% reverse, if 0 TEME2GCRF, if 1 GCRF2TEME

% interp values are from previous init

% outputs
% r_gcrf, 1x3 gcrf position vector
% v_gcrf, 1x3 gcrf velocity vector
% a_gcrf, 1x3 gcrf acceleration vector

if nargin <=8
reverse = 0;
end

% this section is creating time dependent, time quantities,
```

```
% based on Vallado func convtime, just pulled necessary lines for speed
[year,mon,day,hr,min,sec] = invjday(jdk);
utc = hms2sec( hr,min,sec );
tai= utc + d_at;
tt= tai + 32.184;   % sec
[hrtemp,mintemp,sectemp] = sec2hms( tt );
jdtt = jday( year,mon,day, hrtemp, mintemp, sectemp);
ttt= (jdtt - 2451545.0  )/ 36525.0;

% NGA interp for EOP values
[dX,dY] = USNO_EOP_interp(dX_interp,dY_interp,mjd_interp,jdk);

% create prec_nut matrix
[~,~,~,prec_nut] = iau00xys (ttt, dX, dY,iau_data);

% create TEME rotation matrix, see Vallado pp. 234, my method mirros his,
% except uses IAU06-00 to GCRF instead of IAU 75 to J2K
apn = iau_data{7}; apni = iau_data{8};
oblo =  84381.406; % "
ea   =  (((( -0.0000000434 * ttt - 0.000000576 ) * ttt + 0.00200340 )...
* ttt - 0.0001831 ) * ttt -   46.836769 ) * ttt + oblo;
pnsum = 0.0;
[ l, l1, f, d, omega] = fundarg( ttt);
for i = 77 : -1 : 1
tempval = apni(i,1)*l + apni(i,2)*l1 + apni(i,3)*f...
+ apni(i,4)*d + apni(i,5)*omega;
pnsum = pnsum + (apn(i,1) + apn(i,2)*ttt) * sin(tempval) ...
+ (apn(i,5) + apn(i,6)*ttt) * cos(tempval);
end
pplnsum = -0.000135 * pi / (180.0*3600.0);  % " to rad
deltapsi = pnsum + pplnsum;
% equation of the equinoxes developed in IAU-2000 nutation
eqe = deltapsi*cos(ea);

if ~reverse
% perform matrix rotation
r_gcrf = prec_nut*rot3mat(-eqe)*r_sgp4';
v_gcrf = prec_nut*rot3mat(-eqe)*v_sgp4';
else
r_gcrf = rot3mat(-eqe)'*prec_nut'*r_sgp4';
v_gcrf = rot3mat(-eqe)'*prec_nut'*v_sgp4';
end

% transpose to give 1x3
r_gcrf = r_gcrf';
v_gcrf = v_gcrf';

end
```

Space Track TLE Lookup

```
function [TLE] = SpaceTrack_TLE_lookup(norad_id)

% D. Luciani
% Fall 2015

% norad_id = NORAD catalog ID number

url = 'https://www.space-track.org/ajaxauth/login';

options = weboptions('RequestMethod','post','Timeout',20);

data = webread(url,'identity','dluciani@vt.edu','password',...
'VirginiaTechObs','query',['https://www.space-track.org/'...
'basicspacedata/query/class/tle_latest/ORDINAL/1/NORAD_CAT_ID/',...
num2str(norad_id),'/format/tle'],options);

TLE(1,:) = data(1:69);
TLE(2,:) = data(72:140);

end
```

Observable Passes

```
function [visible_passes_local, all_passes_local, tle_epoch_local] = ...
possible_sat_passes(prop_end_time_local, TLE, ...
prop_start_time_local, dt, show_fig,NGA_interp...
d_at,dX_interp,dY_interp,mjd_interp,lod,iau_data)


%% D. Luciani
% dluciani@vt.edu
% Fall 2015

% Function takes in a TLE and a search end time and outputs visible and all
% passes made by the satellite in question over the Blacksburg Observatory.
% Time of interest is TLE epoph to sepcified end time. If no visible passes
% are found, a display output saying so is shown. Note: viewing geometry is
% calulated using nearly circular orbits. Also, there is no year difference
% check between UTC and Local or a check if daylight savings change occurs
% in propagation interval.

% Figure is displayed, showing ground track over time interval in
% question and showing available viewing geometry for the
% Blacksburg Observatory.
```

```
% make sure current folder and all subfolders are in search path
% run in command window: addpath(genpath(pwd)), if need to do this

% all distance units in km, all time unites in seconds, all agles
% measurements in radians, unless otherwise specified


% input:
%
% prop_end_time_local, ending time (local, Eastern Time)

% of search, format: [year, mon, day, hr,min,sec]

% example: prop_end_time = [2015,12,6,23,19,04] local time;
%
% prop_start_time_local, start time (local, Eastern Time)

% of search, format: [year, mon, day, hr,min,sec]

% example: prop_end_time = [2015,12,6,23,19,04] local time;
% note: if not passed in, set to TLE epoch
%
% TLE, Two Line Element from Celestrak or Space-Track (DO NOT
% CHANGE SPACING!)
%TLE =
%['1 25544U 98067A   16041.28784270  .00013572  00000-0  20844-3 0  9999';
%'2 25544  51.6457 335.8524 0006954 101.4053  21.3548 15.54568476985082'];
%
% dt, timestep of orbit evaluation in minutes ,defaul set to 1
% minute
%
% show_fig, a boolean set to 1 (default) if want groundtrack
% plot shown, 0 if not
%     jdk, julian date of time to calcualte interpolaiton at


% dX_interp, Bull. A dX wrt IAU2000A Nutation (msec. of arc),
% dY_interp, Bull. A dY wrt IAU2000A Nutation (msec. of arc),
% Free Core Nutation NOT Removed
% mjd_interp, julian days of update epoch for day

% Inputs are all 3x1 values stored so that linear interpolation can be
% performed. Check julian date of time step versus value epoch to check
% which two values to use for interpolation (done in interpolation
% function)
% NGA_interp, 32x1 matrix of values needed to use NGA interpolating
%%% polynomials at a timestep

% output:   visble_passes_local, list of times of visible passes during
%                 times of interest, in local time, set to null if no
```

```
%                   visible passes
%                 format: [year, mon, day, hr,min,sec] local time
% all_passes_local, list of times of all passes during times of
%                 interest, in local time, set to null if no passes
%                 format: [year, mon, day, hr,min,sec] local time
% tle_epoch_local, tle epoch in [year, mon, day, hr,min,sec] local time

% subfunctions called: jday, twoline2rv, gstime, days2mdh, sgp4, sun,
%                       rot3mat, rot2mat, invjday, sight
%%

% if no input for dt, setting default to 1 min
if nargin<4 || isempty(dt)
dt = 1;
end

% if no input for show_fig, setting default to 1, outputting plot
if nargin<5 || isempty(show_fig)
show_fig = 1;
end

% checking if daylight savings
daylight = isdst(datetime(prop_end_time_local(1:3),'TimeZone',...
'America/New_York'));
if ~daylight
utc2local = -5; % hours from utc to local time
else
utc2local = -4;
end

% convert prop end time from local to utc
prop_end_time_utc = prop_end_time_local - [0 0 0 utc2local 0 0];

% check if next day in utc
if prop_end_time_utc(4)>23
% correct hour
prop_end_time_utc(4) =  prop_end_time_utc(4)-24;
% correct day
prop_end_time_utc(3) =  prop_end_time_utc(3)+1;
end


% Blackburg Observatory:
latgd = 37.222138*pi/180; lon = -80.460020*pi/180+2*pi; %geodetic lattitude
h_msl = .6447; %km

r_site_earth = earth_ellipsoid_radius(latgd);

% converting prop_end_time to julian days
jd_end = jday(prop_end_time_utc(1),prop_end_time_utc(2),...
prop_end_time_utc(3),prop_end_time_utc(4),prop_end_time_utc(5),...
```

```
prop_end_time_utc(6) );

% creating variables to pass to SGP4 initalize
longstr1 = TLE(1,:);
longstr2 = TLE(2,:);
whichconst = 84; % set grav constants to wgs 84 conventions
typerun = 'c'; % set propagation to 'c' = catalog mode (propagates at
%             20 min timesteps from one day before epoch to one day after)
typeinput = 0; % initialized, only needed for manual mode

% initialize for SGP4
% twoline2rv calls sgp4init
[satrec] = twoline2rv(whichconst, longstr1, ...
longstr2, typerun,typeinput);

% calculate epoch year of TLE
year = 2000+satrec.epochyr;

% convert to epoch date
[mon,day,hr,minute,sec] = days2mdh ( year,satrec.epochdays);

% check if utc to local changes day and corrects
if hr+utc2local<0;
hr = hr+24;
day = day-1;
end

% create date for TLE epoch in local and utc
tle_epoch_local = [year,mon,day,hr+utc2local,minute,sec]; % local
tle_epoch_utc = [year,mon,day,hr,minute,sec];

% if prop start time is not passed in
if nargin < 3 || isempty(prop_start_time_local)

% set epoch to tle epoch
jde = satrec.jdsatepoch;

% mins from tle epoch to start epoch
epoch_offset = 0;

% if prop start time is passed in
else
% calculate start time in utc
prop_start_time_utc = prop_start_time_local - [0 0 0 utc2local 0 0];

% check if next day in utc
if prop_start_time_utc(4)>23
% correct hour
prop_start_time_utc(4) =  prop_start_time_utc(4)-24;
% correct day
prop_start_time_utc(3) =  prop_start_time_utc(3)+1;
```

```
end

% calculat julian day of start time
jde = jday(prop_start_time_utc(1),prop_start_time_utc(2),...
prop_start_time_utc(3),prop_start_time_utc(4),...
prop_start_time_utc(5),prop_start_time_utc(6) );

% mins from tle epoch to start epoch
epoch_offset = jde*24*60-satrec.jdsatepoch*24*60;

end

% disp warning if backwards propagation
if epoch_offset < 0
disp('Note: prop start time before tle epoch, may cause inaccuracy')
end


% make sure prop start time is before prop end time
if jde>jd_end
if jde == satrec.jdsatepoch
disp('Error: propagation end time is before TLE epoch')
visible_passes_local = []; all_passes_local = [];
return
else
disp('Error: propatation start time is after end time')
visible_passes_local = []; all_passes_local = [];
return
end
end

%% call sgp4

% create time vector of minues elapsed from TLE epoch to ending time
% starting at 1 minute from TLE epoch with a timestep of dt minute
t_vec = 0:dt:jd_end*24*60-jde*24*60; % minutes

% setting minimum viewing elevation of observatory, set to 10 degrees
el_min = 10*pi/180;

% initialize counting variable used in for loop
k = 1; % num all passes = k-1
j = 1; % num vis passes = j-1

% for loop runs through time vector
for i = 1:length(t_vec)

% create julian date for timestep
jdk = jde+t_vec(i)./60./24;

% call sgp4 for specified time
```

```
[~, r_sgp4, v_sgp4] = sgp4(satrec,t_vec(i)+epoch_offset);
% r and v in km and km/s
[r_gcrf,v_gcrf] = sgp4_teme2gcrf(r_sgp4,v_sgp4,jdk,...
d_at,dX_interp,dY_interp,mjd_interp,iau_data);

% viewing geometry for circular orbit, forulation in Wertz, "Orbit and
% Constellation Design and Management"
% calculated inside of loop because it is a function of current height,
% which can vary signficantly if orbit is not nearly circular
rho = asin(r_site_earth/(norm(r_gcrf)));

nu_max = asin(sin(rho)*cos(el_min));

% radius of available viewing circle
lam_max = pi/2-el_min-nu_max;

% find min and max of available viewing long
lon_min = lon-lam_max; lon_max = lon+lam_max;

% calculate viewing geometry circle equations
% upper half of circle
yp = @(x) (lam_max^2-(x-lon).^2).^(1/2)+latgd;

% lower half of circle
ym = @(x) -(lam_max^2-(x-lon).^2).^(1/2)+latgd;

% implementint iau-06/00 cio approach
[r_itrf] = gcrf2itrf(r_gcrf,v_gcrf,[],jdk,NGA_interp,...
dX_interp,dY_interp,mjd_interp,d_at,lod,iau_data);

% lat,long function
[lat(i), long(i)] = sub_satellite_point(r_itrf);

% check if lat,long of satellite is within viewing geometry
if long(i)>=lon_min && long(i)<=lon_max && lat(i)>=...
ym(long(i)) && lat(i)<=yp(long(i)) % circle search area

% store time for pass
store_allt(k) = t_vec(i);

% increment all pass counting variable
k = k+1;

% calculate sun vector, output is in eci frame with au (units)
[rsun_au] = sun ( jde+t_vec(i)./60./24 );

% convert au to km
rsun = rsun_au.*149597870.700;

% create eci to sez (south, east, zenith) rotation matrix
% note: polar motion, nutation, and precession not accounted for
```

```
rsun_itrf = gcrf2itrf(rsun,[],[],jdk,NGA_interp,dX_interp,...
dY_interp,mjd_interp,d_at,lod,iau_data);
Rsez_itrf = rot2mat(pi/2-latgd)*rot3mat(lon);

% calculate sun vector in sez
rsun_sez = Rsez_itrf*rsun_itrf';

rsun_sez = rsun_sez'-[0 0 r_site_earth+h_msl];

% calculate sun elevation angle (sez is local sky frame)
el_sun = asin(rsun_sez(3)/norm(rsun_sez));

% check if line of sight vector exists between satellite and sun,
% ie if earth is in between sun or not
% los is a boolean, 'e' uses ellipsoid model of earth
% uses "shadow cylinder" behind earth, see Vallado
[los] = sight ( rsun,r_gcrf);

% checks if there is a line of sight and if the sun is set below
% -12
% degrees elevation (see Vallado for illumination versus elevation
% plot)
if los == 1 && el_sun < -12*pi/180

% store visible pass time
store_terminatorr(j) = t_vec(i);

% store time index of visible pass in vector
store_i(j) = i;

% increment visible pass counter
j = j+1;

end

end

end

% check if there are any passes
if k>1

% convert all passes from minutes elapsed to julian days
store_alltt = jde+store_allt./60./24; % time vec in jdays

% for all passes
for i = 1:length(store_alltt)

% convert from jday to date
[year,mon,day,hr,min,sec] = invjday(store_alltt(i));
```

```
% check if day change between utc and local and change if necessary
if hr+utc2local<0;
hr = hr+24;
day = day-1;
end

% calcualte all passes in local time
all_passes_local(i,:) = [year,mon,day,hr+utc2local,min,sec];
% local blacksbrug time

end

else % if no passes
all_passes_local = [];
disp('No passes.')
end

% check if any visible passes
if j>1

% convert visible passes from minutes elapsed to julian days
store_terminator = jde+store_terminatorr./60./24; % time vec in jdays

% for visible passes
for i = 1:length(store_terminator)

% julian days to date
[year,mon,day,hr,min,sec] = invjday(store_terminator(i));

% check if day change between utc and local and change if necessary
if hr+utc2local<0;
hr = hr+24;
day = day-1;
end

% calculate visible passes in local time
visible_passes_local(i,:) = [year,mon,day,hr+utc2local,min,sec];
% local blacksbrug time

end

else % no visible passes
visible_passes_local = [];
disp('No visible passes.')
end

% calcualte max and min longitude
xrange_max = lon+lam_max;
xrange_min = lon-lam_max;

% create longitude vector
```

```
x_vec = linspace(xrange_min,xrange_max,10000);

% create points to graph (in degrees)
y = [yp(x_vec) ym(x_vec)].*180./pi;
x = [x_vec x_vec].*180./pi;

% if want to show plot
if show_fig

% create new figure
figure
% Plot Earth coastline in the background
load topo
contour(0:359,-89:90,topo,[0 0],'b')
axis equal, box on, grid on
title2 = title({['\makebox[4in][c]{NORAD ID: ',num2str(...
satrec.satnum),'}'];['\makebox[4in][c]{',datestr(...
prop_start_time_local),' to ',datestr(prop_end_time_local)...
,' Eastern Time}']});
xl = xlabel('Longitude ($^\circ$)');
yl = ylabel('Latitude ($^\circ$)');
set([title2 xl yl],'interpreter','latex','fontsize',10)
set(gca,'XLim',[180 360],'YLim',[-90 90],'XTick',[180:60:360],...
'Ytick',[-90:30:90]);
% set(gca,'XLim',[0 360],'YLim',[-90 90],'XTick',[0:60:360],...
%     'Ytick',[-90:30:90]);

%
hold on

% converts to deg and makes column matrix
long = (long.*180/pi)';
lat = (lat.*180/pi)';

% if visible passes
if j>1

% running through all long,lat pairs
for i = 1:1:length(long)

% if long,lat pair is a visible pass, plot in green, if not in
% black
if any(store_i == i)
plot(long(i), lat(i),'.','Color','g','MarkerSize',7)
else
plot(long(i), lat(i),'.','Color','k','MarkerSize',7);
end
end

% no visible passes
else
```

```
% plots lat,long of satellite in all black dots
hold on
plot(long, lat,'.','Color','k','MarkerSize',7)
title({['Ground Track']; ['NORAD ID: ',num2str(satrec.satnum)];
[datestr(prop_start_time_local),' to ',datestr(...
prop_end_time_local),' Eastern Time']})
xlabel('Longitude (\circ)')
ylabel('Latitude (\circ)')

end

% plots viewing geometry circle
hold on
plot(x_vec.*180./pi, yp(x_vec).*180./pi,x_vec.*180./pi, ym(x_vec).*...
180./pi,'Color','r','MarkerSize',2)

end % end ground track figure plotting

end % end function
```

NGA Earth Orientation Parameters Lookup

```
function [NGA_interp,d_at] = ...
NGA_EOP_lookup_and_interp_init(time_view_local,lt)

% D. P. Luciani
% dluciani@vt.edu
% Fall 2015

% NOTE: cannot look up for a pass "tonight" until after
% update at about 1700 UTC
% (1300 edt, 1200 est) if want to look at long term, lookup for an epoch =
% time_view_local - 1 day, but will be outside interpolation and accuracy
% will suffer.

% This function looks up the required, NGA, Earth Orientation
% Parameter (EOP) values
% needed to linearly interpolate to get values for IAU-2006/2000 GCRF to
% ITRF reduction formulation (CIO approach)

% NGA daily usually updates around 17:00 UTC, daily epoch is 00:00 UTC

% inputs:

% time_view_local, local date of approx intended view,
% format [year mon day hr min sec]
% lt, long term propagation?, 0 if false, 1 if true, default set to 0
```

```
if nargin<2
lt = 0;
end
if lt == 1
c = zeros(1,6); c(1,3) = -1;
time_view_local = datevec(now)+c;
end

% outputs:

% d_at, TAI-UTC difference in seconds (will not change over pass, so only
%%% need to find for init
% NGA_interp, 32x1 matrix of values needed to use NGA interpolating
%%% polynomials at a timestep

% check if night view
if time_view_local(4)>12
nightview = 1;
else
nightview = 0;
end

% testing func
% clear all; clc;
% date_of_view_local = [16,2,24]; nightview = 1;

% convert local view time to function specific time format
date_local_func(1) = time_view_local(1)-2000;
date_local_func(2) = time_view_local(2);
date_local_func(3) = time_view_local(3);

% if it is a night viewing, closest utc day is day+1, if it is a morning
% view closest utc day is same day
if nightview
day_utc = date_local_func(3)+1;
else
day_utc = date_local_func(3);
end

% creating variabls to call NGA website
y = num2str(date_local_func(1));
y = y(2);
year = 2000+date_local_func(1);
days = day(datetime(year,date_local_func(2),day_utc),'dayofyear');
days = num2str(days);

% create zeros if needed for ddd string format
if numel(days)==1
days = ['00' days];
elseif numel(days)==2
```

```
days = ['0' days];
end

% create url
url = ['ftp://ftp.nga.mil/pub2/gps/eopp/2016eopp/EOPP',y,days,'.TXT'];

% NGA lookup
data = urlread(url);

% create cell of text in columns
% make sure to disable whitespace so textscan catches spaces at beginning
% of line!
data = textscan(data,'%s','Delimiter','endofline','whitespace','');

% decompose into rows easiest because row length not all same..
data_row1 = cell2mat(data{1,1}(1,:));
data_row2 = cell2mat(data{1,1}(2,:));
data_row3 = cell2mat(data{1,1}(3,:));
data_row4 = cell2mat(data{1,1}(4,:));
data_row5 = cell2mat(data{1,1}(5,:));
d_at = str2double(data_row5(1:4));

% init
NGA_interp = zeros(1,32);

% extract values, syntax found at:
% http://earth-info.nga.mil/GandG/sathtml/eoppdoc.html
NGA_interp(1) = str2double(data_row1(1:10));
NGA_interp(2) = str2double(data_row1(11:20));
NGA_interp(3) = str2double(data_row1(21:30));
NGA_interp(4) = str2double(data_row1(31:40));
NGA_interp(5) = str2double(data_row1(41:50));
NGA_interp(6) = str2double(data_row1(51:60));
NGA_interp(7) = str2double(data_row1(61:70));
NGA_interp(8) = str2double(data_row1(71:end));
NGA_interp(9) = str2double(data_row2(1:6));
NGA_interp(10) = str2double(data_row2(7:16));
NGA_interp(11) = str2double(data_row2(17:26));
NGA_interp(12) = str2double(data_row2(27:36));
NGA_interp(13) = str2double(data_row2(37:46));
NGA_interp(14) = str2double(data_row2(47:56));
NGA_interp(15) = str2double(data_row2(57:66));
NGA_interp(16) = str2double(data_row2(67:72));
NGA_interp(17) = str2double(data_row2(73:end));
NGA_interp(18) = str2double(data_row3(1:10));
NGA_interp(19) = str2double(data_row3(11:20));
NGA_interp(20) = str2double(data_row3(21:30));
NGA_interp(21) = str2double(data_row3(31:40));
NGA_interp(22) = str2double(data_row3(41:50));
NGA_interp(23) = str2double(data_row3(51:60));
NGA_interp(24) = str2double(data_row3(61:end));
```

```
NGA_interp(25) = str2double(data_row4(1:10));
NGA_interp(26) = str2double(data_row4(11:20));
NGA_interp(27) = str2double(data_row4(21:30));
NGA_interp(28) = str2double(data_row4(31:40));
NGA_interp(29) = str2double(data_row4(41:49));
NGA_interp(30) = str2double(data_row4(50:58));
NGA_interp(31) = str2double(data_row4(59:67));
NGA_interp(32) = str2double(data_row4(68:end));

end
```

NGA Earth Orientation Parameters Interpolation

```
function [xpm,ypm,dut1] = NGA_EOP_interp(NGA_interp,jdk)

% D. P. Luciani
% dluciani@vt.edu
% Fall 2015

% Function interpolates USNO EOP values to get value for desired timestep

% inputs:
% jdk, julian date of time to calcualte interpolaiton at
% NGA_interp, 32x1 matrix of values needed to use NGA interpolating
%%% polynomials at a timestep

% outputs:
% xpm, x polar motion at timestep, in rad
% ypm, y polar motion at timestep, in rad
% dut1, ut1-utc at timestep, in seconds


% timestep from jd to mjd
t = jdk-2400000.5;

ta = NGA_interp(1); A = NGA_interp(2); B = NGA_interp(3);
C = NGA_interp(4:5); D = NGA_interp(6:7);
P = NGA_interp(8:9); E = NGA_interp(10); F = NGA_interp(11);
G = NGA_interp(12:13); H = NGA_interp(14:15);
Q = NGA_interp(16:17); tb = NGA_interp(18); I = NGA_interp(19);
J = NGA_interp(20); K = NGA_interp(21:24);
L = NGA_interp(25:28); R = NGA_interp(29:32);

% interpolation formulas found at:
% http://earth-info.nga.mil/GandG/sathtml/eoppdoc.html
% note matrix mult for summation

xpm = A+B*(t-ta)+C*sin(2*pi*(t-ta)./P')+D*cos(2*pi*(t-ta)./P');
```

```
% as to rad
xpm = xpm/3600*pi/180;

ypm = E+F*(t-ta)+G*sin(2*pi*(t-ta)./Q')+H*cos(2*pi*(t-ta)./Q');
% as to rad
ypm = ypm/3600*pi/180;

dut1 = I+J*(t-tb)+K*sin(2*pi*(t-tb)./R')+L*cos(2*pi*(t-tb)./R');

end
```

USNO Earth Orientation Parameters Lookup

```
function [dX_interp,dY_interp,mjd_interp,lod] = ...
USNO_EOP_lookup_and_interp_init(time_view_local)

% D. P. Luciani
% dluciani@vt.edu
% Fall 2015

% This function looks up the required, U.S. Naval Observatory (USNO)
% published (2000A finals daily), Earth Orientation Parameter (EOP) values
% needed to linearly interpolate to get values for IAU-2006/2000 GCRF to
% ITRF reduction formulation (CIO approach)

% want to call before pass calculation loop.

% USNO daily usually updates around 17:00 UTC, daily epoch is 00:00 UTC

% inputs:

% time_view_local, local date of view, format [year mon day hr min sec]

% outputs (see http://maia.usno.navy.mil/ser7/readme.finals2000A):

% USE NGA EOP xp, Bull. A PM-x (sec. of arc)
% USE NGA EOP yp, Bull. A PM-x (sec. of arc)
% USE NGA EOP dut1, Bull. A UT1-UTC (sec. of time)

% dX, Bull. A dX wrt IAU2000A Nutation (msec. of arc),
% dY, Bull. A dY wrt IAU2000A Nutation (msec. of arc), ]
% Free Core Nutation NOT Removed
% mjd, modified julian days of update epoch for day
% lod, extra length of day, in sec, estimated from pevious utc, becuase
%%% value not available at current time, does not change with time so only
%%% needs to be called during init
```

```
% Outputs are all 3x1 values stored so that linear interpolation can be
% performed. Check julian date of time step versus value epoch to check
% which two values to use for interpolation (done in interpolation
% function)

% check if night view
if time_view_local(4)>12
nightview = 1;
else
nightview = 0;
end

% convert local view time to function specific time format
date_local_func(1) = time_view_local(1)-2000;
date_local_func(2) = time_view_local(2);
date_local_func(3) = time_view_local(3);

% test function
% clear all; clc;
% jdk = 2457443.6;
% date_view_local = [16,2,25];
% nightview = 1;

% online lookup
data = webread('http://maia.usno.navy.mil/ser7/finals2000A.daily');

% converts data to matrix
data = textscan(data,'%s','Delimiter','endofline');
M = cell2mat(data{1});

% extract date from data line and covert to number
year = str2num(M(:,1:2));
month = str2num(M(:,3:4));
day = str2num(M(:,5:6));

% find index for date where year month and date match given date
one = year == date_local_func(1);
two = month == date_local_func(2);
three = day == date_local_func(3);
holder = one+two+three;
ind1 = find(holder == 3);
% hold = [year month day];

% create additional idecies for two next days, to have if needed to
% interpolate
% i.e. if morning view want utc date = local date and local date + 1 for
% interpolation, if night view before 00:00 utc want utc = local date and
% local date + 1, if equal to utc want utc = local date + 1, if after
% 00:00 utc want utc = local date + 1 and local date + 2 for interpolation
ind2 = ind1+1;
ind3 = ind2+1;
```

```
% pull pertinent data from data matrix
data_line = M(ind1:ind3,:);

% if morning view need ind1-2, becuase updates after morning
if nightview
lod_ind = ind1-1;
else
lod_ind = ind1-2;
end

% lod pull from last available
lod = str2double(M(lod_ind,80:86));
% ms to sec
lod = lod/1000;

% extract needed values and convert to numbers
mjd = data_line(:,8:15);
% xp = data_line(:,19:27);
% yp = data_line(:,38:46);
% dut1 = data_line(:,59:68);
dX_interp = data_line(:,98:106);
dY_interp = data_line(:,117:125);

mjd_interp = str2num(mjd); % convert from mjd to jd
% xp = str2num(xp);
% yp = str2num(yp);
% dut1 = str2num(dut1);
dX_interp = str2num(dX_interp);
dY_interp = str2num(dY_interp);

end
```

USNO Earth Orientation Parameters Interpolation

```
function [dX,dY] = USNO_EOP_interp(dX_interp,dY_interp,mjd_interp,jdk)

% D. P Luciani
% dluciani@vt.edu
% Fall 2015

% Function interpolates USNO EOP values to get value for desired timestep

% inputs:
% jdk, julian date of time to calcualte interpolaiton at

% dX_interp, Bull. A dX wrt IAU2000A Nutation (msec. of arc),
% dY_interp, Bull. A dY wrt IAU2000A Nutation (msec. of arc),
```

```
% Free Core Nutation NOT Removed
% mjd_interp, julian days of update epoch for day

% Inputs are all 3x1 values stored so that linear interpolation can be
% performed. Check julian date of time step versus value epoch to check
% which two values to use for interpolation (done in interpolation
% function)

% outputs:

% dX, interpolated, in rad
% dY, interpolated, in rad

% convert timestep to mjd to avoid precision errors
mjdk = jd2mjd(jdk);

% check if jd timestep is exactly equal to the middle jd, possible
% but unlikely
if mjdk == mjd_interp(2)
dX = dX_interp(2);
dY = dY_interp(2);
return
end

% if jd timestep is greater than jd middle than want to use second and
% third values for interpolation, else the first and second values
if mjdk>mjd_interp(2)
ind1 = 2; ind2 = 3;
else
ind1 = 1; ind2 = 2;
end

% create linear interpolation
% shift jds such that jd(ind1) is at x = 0
% do so by subtracting jd(ind1)
m = (dX_interp(ind2)-dX_interp(ind1))/(mjd_interp(ind2)-mjd_interp(ind1));
b = dX_interp(ind1);

dX = m*(mjdk-mjd_interp(ind1))+b;
% msec of arc to rad
dX = dX/1000/3600*pi/180;

m = (dY_interp(ind2)-dY_interp(ind1))/(mjd_interp(ind2)-mjd_interp(ind1));
b = dY_interp(ind1);

dY = m*(mjdk-mjd_interp(ind1))+b;
% msec of arc to rad
dY = dY/1000/3600*pi/180;

end
```

Camera Control Initialization

```
function [cam,connected] = cam_control_init()

cam = actxserver('IC.ICImagingControl3.1');

try
cam.device = ('DMx 41BU02');
catch
connected = 0;
disp('Camera failed to connect, recheck connection.')
return
end
connected = 1;

% set parameters
try
cam.Exposure = 2; % 4 seconds
catch
cam.ExposureAuto = 0;
cam.Exposure = 2;% 4 seconds
end

cam.Brightness = 0;

try
cam.Gain = 1000;
catch
cam.GainAuto = 0;
cam.Gain = 1000;
end

cam.Gamma = 20;

end
```

Mount Control Initialization

```
function [mount_object,connected] = mount_control_init()

% for Telescope Simulator
% mount_object = actxserver('ScopeSim.Telescope');

% for Prices Fork Mount
mount_object = actxserver('ASCOM.Celestron.Telescope');
```

```
try
mount_object.connected = 1;
catch
connected = 0;
disp('Mount failed to connect, recheck connection.')
return
end
connected = 1;
end
```

Surveillance User Input

```
function [pointing_data_cell,date_val] = nightly_tasking_user_input()

% USER INPUT:
% current (viewing) date
% norad_id
% TLE
% pass_start_time
% pass_end_time

% ouput: pointing_data_cell = [norad_id', TLE', t_pass, tra_dechrs,
% tdec_deg,  az_deg, el_deg, tle_epoch_local,
% pass_start_time', pass_end_time'];

% one line for each pass

% set to current night of tasking
date_val = [2016,4,14,0,0,0]; % [yr,mo,day,0 hr,0 min,0 sec] current date

% note these are all cells

% first pass (only first pass commented, all others follow same template)
% pass order number for night
pass_num = 1;

% norad_id of satellite for pass
norad_id{pass_num} = 23343;

% You can input the TLE or look it up on SpaceTrack using this function
% TLE{pass_num} = ['1 25544U 98067A   16048.72897402  .00011132
% 00000-0  17503-3 0  9995';
% '2 25544  51.6323 298.7205 0011080 116.7902 280.2695 15.54100352986248'];
TLE{pass_num} = SpaceTrack_TLE_lookup(norad_id{pass_num});

% LOCAL pass time parameters from possible_sat_passes.m or HeavensAbove
% time of pass start in [yr,mo,day,hr,min,sec]
% note only need to add hour min seconds to current date
```

```
pass_start_time{pass_num} = date_val+[0,0,0,20,29,14];

% LOCAL time of pass end in [yr,mo,day,hr,min,sec]
pass_end_time{pass_num} = date_val+[0,0,0,20,37,42];


% call EOP and iau inits - only need to init for the first pass
% DON'T CHANGE THIS
init_data = EOP_IAU_init(pass_start_time{1});


% run surveillance function to find closest approach time and angles
% DON'T CHANGE THIS
[surv_data_cell(pass_num,:)] = surveillance_function(TLE{pass_num}, ...
pass_start_time{pass_num}, pass_end_time{pass_num}, init_data);


% second pass
pass_num = 2;

% change:
norad_id{pass_num} = 15494;
TLE{pass_num} = SpaceTrack_TLE_lookup(norad_id{pass_num});
pass_start_time{pass_num} = date_val+[0,0,0,20,36,22];
pass_end_time{pass_num} = date_val+[0,0,0,20,43,50];

% don't change:
[surv_data_cell(pass_num,:)] = surveillance_function(TLE{pass_num}, ...
pass_start_time{pass_num}, pass_end_time{pass_num}, init_data);

% third pass
pass_num = 3;
norad_id{pass_num} = 25544;
TLE{pass_num} = SpaceTrack_TLE_lookup(norad_id{pass_num});

pass_start_time{pass_num} = date_val+[0,0,0,20,44,17];
pass_end_time{pass_num} = date_val+[0,0,0,20,48,44];

[surv_data_cell(pass_num,:)] = surveillance_function(TLE{pass_num}, ...
pass_start_time{pass_num}, pass_end_time{pass_num}, init_data);

% fourth pass
% pass_num = 4;
% norad_id{pass_num} = 23343;
% TLE{pass_num} = SpaceTrack_TLE_lookup(norad_id{pass_num});
%
% pass_start_time{pass_num} = date_val+[0,0,0,20,28,42];
% pass_end_time{pass_num} = date_val+[0,0,0,20,28,42];
%
% [surv_data_cell(pass_num,:)] = surveillance_function(TLE{pass_num}, ...
%     pass_start_time{pass_num}, pass_end_time{pass_num}, init_data);
```

```
% up to n passes (make sure passes don't overlap!)

% create pass data cell
pointing_data_cell = [norad_id', TLE', surv_data_cell, ...
pass_start_time', pass_end_time'];

end
```

Surveillance Task Calculation

```
function [surv_data_cell] = surveillance_function(TLE, ...
pass_start_time_local, pass_end_time_local, init_data)

% D. Luciani
% dluciani@vt.edu
% Spring 2015

NGA_interp = init_data{1,1}; d_at = init_data{1,2};
dX_interp = init_data{1,3}; dY_interp = init_data{1,4};
mjd_interp = init_data{1,5}; lod = init_data{1,6};
iau_data = init_data{1,7};


% all distance units in km, all time unites in seconds, all agles
% measurements in radians, unless otherwise specified


% input:

% pass_end_time_local, ending time (local, Eastern Time)
%             of search, format: [year, mon, day, hr,min,sec]
%               example: prop_end_time = [2015,12,6,23,19,04] local time;

%           pass_start_time_local, start time (local, Eastern Time)
%             of search, format: [year, mon, day, hr,min,sec]
%               example: prop_end_time = [2015,12,6,23,19,04] local time;
%           note: if not passed in, set to TLE epoch

%           TLE, Two Line Element from Celestrak or Space-Track (DO NOT
%                 CHANGE SPACING!)
% TLE = ['1 25544U 98067A   16041.28784270  .00013572
% 00000-0  20844-3 0  9999';
% '2 25544  51.6457 335.8524 0006954 101.4053  21.3548 15.54568476985082'];

% init_data, EOP init
```

```
%%% polynomials at a timestep

% output:   parameters for minimum range during pass
% surv_data_cell = {t_pass, tra_dechrs, tdec_deg,
% az_deg, el_deg, tle_epoch_local};

dt = .01; % min

% checking if daylight savings
daylight = isdst(datetime(pass_end_time_local(1:3)...
,'TimeZone','America/New_York'));
if ~daylight
utc2local = -5; % hours from utc to local time
else
utc2local = -4;
end

% convert prop end time from local to utc
prop_end_time_utc = pass_end_time_local - [0 0 0 utc2local 0 0];

% check if next day in utc
if prop_end_time_utc(4)>23
% correct hour
prop_end_time_utc(4) =  prop_end_time_utc(4)-24;
% correct day
prop_end_time_utc(3) =  prop_end_time_utc(3)+1;
end


% Blackburg Observatory:
latgd = 37.222138*pi/180; lon = -80.460020*pi/180+2*pi;
% geodetic lattitude
h_msl = .6447; %km

r_site_earth = earth_ellipsoid_radius(latgd);

% converting prop_end_time to julian days
jd_end = jday(prop_end_time_utc(1),prop_end_time_utc(2),...
prop_end_time_utc(3),prop_end_time_utc(4),prop_end_time_utc(5)...
,prop_end_time_utc(6) );

% creating variables to pass to SGP4 initalize
longstr1 = TLE(1,:);
longstr2 = TLE(2,:);
whichconst = 84; % set grav constants to wgs 84 conventions
typerun = 'c'; % set propagation to 'c' = catalog mode (propagates
%       at 20 min timesteps from one day before epoch to one day after)
typeinput = 0; % initialized, only needed for manual mode

% initialize for SGP4
```

```
% twoline2rv calls sgp4init
[satrec] = twoline2rv(whichconst, longstr1, longstr2, typerun,typeinput);

% calculate epoch year of TLE
year = 2000+satrec.epochyr;

% convert to epoch date
[mon,day,hr,minute,sec] = days2mdh ( year,satrec.epochdays);

% check if utc to local changes day and corrects
if hr+utc2local<0;
hr = hr+24;
day = day-1;
end

% create date for TLE epoch in local and utc
tle_epoch_local = [year,mon,day,hr+utc2local,minute,sec]; % local
tle_epoch_utc = [year,mon,day,hr,minute,sec];

% if prop start time is not passed in
if nargin < 3 || isempty(pass_start_time_local)

% set epoch to tle epoch
jde = satrec.jdsatepoch;

% mins from tle epoch to start epoch
epoch_offset = 0;

% if prop start time is passed in
else
% calculate start time in utc
prop_start_time_utc = pass_start_time_local - [0 0 0 utc2local 0 0];

% check if next day in utc
if prop_start_time_utc(4)>23
% correct hour
prop_start_time_utc(4) =  prop_start_time_utc(4)-24;
% correct day
prop_start_time_utc(3) =  prop_start_time_utc(3)+1;
end

% calculat julian day of start time
jde = jday(prop_start_time_utc(1),prop_start_time_utc(2),...
prop_start_time_utc(3),prop_start_time_utc(4),...
prop_start_time_utc(5),prop_start_time_utc(6) );

% mins from tle epoch to start epoch
epoch_offset = jde*24*60-satrec.jdsatepoch*24*60;

end
```

```
% disp warning if backwards propagation
if epoch_offset < 0
disp('Propagation start time before tle epoch, may cause inaccuracy')
end


% make sure prop start time is before prop end time
if jde>jd_end
if jde == satrec.jdsatepoch
disp('Error: propagation end time is before TLE epoch')
return
else
disp('Error: propatation start time is after end time')
return
end
end

%% call sgp4

% create time vector of minues elapsed from TLE epoch to ending time
% starting at TLE epoch with a timestep of dt minute
t_vec = 0:dt:jd_end*24*60-jde*24*60; % minutes

time_v = jde+t_vec./60./24;
for i = 1:length(time_v)

% julian days to date
[year,mon,day,hr,mins,sec] = invjday(time_v(i));

% check if day change between utc and local and change if necessary
if hr+utc2local<0;
hr = hr+24;
day = day-1;
end

% calculate visible passes in local time
time_vec_local(i,:) = [year,mon,day,hr+utc2local,mins,sec];
% local blacksbrug time

end
for i = 1:length(time_v)

% julian days to date
[year,mon,day,hr,mins,sec] = invjday(time_v(i));

% calculate visible passes in local time
time_vec_utc(i,:) = [year,mon,day,hr,mins,sec]; % local blacksbrug time

end

% for loop runs through time vector
```

```
for i = 1:length(t_vec)

% create julian date for timestep
jdk = jde+t_vec(i)./60./24;

% call sgp4 for specified time
[~, r_sgp4, v_sgp4] = sgp4(satrec,t_vec(i)+epoch_offset);
% r and v in km and km/s
[r_gcrf,v_gcrf] = sgp4_teme2gcrf(r_sgp4,v_sgp4,jdk,...
d_at,dX_interp,dY_interp,mjd_interp,iau_data);

% implementint iau-06/00 cio approach
[r_itrf,v_itrf] = gcrf2itrf(r_gcrf,v_gcrf,[],jdk,NGA_interp...
,dX_interp,dY_interp,mjd_interp,d_at,lod,iau_data);

% calculating range, topo ra and dec, and az and el for each timestep
[rho(i),tra(i),tdec(i)] = rv2tradc ( r_gcrf(:)',v_gcrf(:)',jdk,...
NGA_interp,dX_interp,dY_interp,mjd_interp,d_at,lod,iau_data);
[rhoecef(i),az(i),el(i)] = rv2razel ( r_itrf(:)',v_itrf(:)');

end

ind = find(min(rho)==rho);
t_pass = time_vec_local(ind,:);
az_deg = az(ind)*180/pi;
el_deg = el(ind)*180/pi;

tra_dechrs = tra(ind)*180/pi/15;
tdec_deg = tdec(ind)*180/pi;

surv_data_cell = {t_pass, tra_dechrs, tdec_deg, az_deg, el_deg, ...
tle_epoch_local};

end % end function
```

Surveillance Operation Code

```
% Surveillance Operation

% RUN SECTION BY SECTION (CTRL-ENTER with cursor in section)

% D. Luciani

% dluciani@vt.edu
% Sring 2015

% Section 1
```

```
clearvars; clc;

warning('off','all')

% make sure 'telescope control' is the current path
try
check_path()
catch
disp('Set the current path to the "telescope control" folder.')
break
end

% add all subfolders to path
addpath(genpath(pwd))

section1_run = 1;

%% Section 2

% MAKE SURE YOU HAVE UPDATED nightly_tasking_user_input.m with the pass
% information

% this section could take a few minutes, depending on the number of
% satellites

% you need a internet connection for this section

% Pass parameters

% check to make sure sections are run in order
if ~exist('section1_run','var')
disp('You need to run Section 1 before running this section.')
break
end

[pointing_data_cell,date_val] = nightly_tasking_user_input();

if any(date_val ~= datevec(date))
disp('Update the tasking input!')
break
end

% save pointing data for image post-processing
dt = datestr(pointing_data_cell{1,9},'yy_mm_dd');
dt = [pwd '\collected_data\' dt '_pointing_data.mat'];
save(dt,'pointing_data_cell')

section2_run = 1;

% save data in case MATLAB gets interupted when going transporting to
% observatory
```

```
save info.mat pointing_data_cell section2_run


%% Section 3
% Run this section after the hardware setup is complete, you have
% initialized the telescope, and all connections to the control computer
% are connected

% check to make sure sections are run in order
if ~exist('section2_run','var')
disp('You need to run Section 3 before running this section.')
break
end

load info.mat % make sure this is the data you saved

% init cam and mount objects
[mount,mount_connect] = mount_control_init();

[cam,cam_connect] = cam_control_init();

% You need to focus the camera now, point at a brighter star and
% focus to get that star as close to a point in the image as possible
% (note the exposure is set to 4 seconds so any focus change will take 4
% seconds to show up)
disp('Focus the camera.')
cam.LiveStart

% if the image is not showing any stars after focusing, (should not
% normally be the case) you may have to change camera paramters, uncomment:

% cam.ShowPropertyDialog

% and change the exposure time, if imaging near sunset may need to reduce
% the expousre time, if there are still issues you may have to alter gamma
% on the second tab. Once you are satisfied with the image click apply,
% update, and ok. Then run next section.


section3_run = 1;

%% Section 4a -- Run once for all satellite passes

% BEFORE RUNNING THIS SECTION MAKE SURE YOU ARE READY FOR TELESCOPE TO MOVE

% Run this section when you are ready to start automatic image collection.

% MATLAB will "pause" when wating for next pass, do not interrupt or
% function will have an error (function may lock up MATLAB for a while,
% depending on the time difference in the passes specified by TLEs or norad
% IDs, you can do other things on computer such as look at last round of
```

```
% images)

% check to make sure sections are run in order
if ~exist('section3_run','var') || ~mount_connect || ~cam_connect
disp(['You need to run Section 3 and make sure hardware'...
'is connected before running this section.'])
break
end

cam.LiveStop

[done] = surveillance_task(pointing_data_cell, mount, cam)

%% Section 4b -- Run once per satellite (or if automated gets interupted)

% BEFORE RUNNING THIS SECTION MAKE SURE YOU ARE READY FOR TELESCOPE TO MOVE

% Run this section when you are ready to start automatic image collection.

% MATLAB will "pause" while wating for satellite pass when you run this
% section, do not interrupt or function will have an error
% (i.e. wait until you are ready to run this section)

% check to make sure sections are run in order
if ~exist('section3_run','var') || ~mount_conncet || ~cam_connect
disp(['You need to run Section 3 and make sure hardware'...
'is connected before running this section.'])
break
end

cam.LiveStop

% USER INPUT which satellite in task list do you wish to task
sat = 2;

[done] = surveillance_task_single(pointing_data_cell, mount, cam, sat)
```

Surveillance Tasking Function

```
function [done] = surveillance_task(pointing_data_cell, mount, cam)

% D. Luciani

% dluciani@vt.edu
% Spring 2015

tra_hrs = cell2mat(pointing_data_cell(:,4));
tra_hrs(tra_hrs<0) = tra_hrs(tra_hrs<0)+24;
```

```
t_pass = cell2mat(pointing_data_cell(:,3));
tdec_deg = cell2mat(pointing_data_cell(:,5));
norad_id = cell2mat(pointing_data_cell(:,1));

d2s = 86400; % seconds per day

disp('Operating...')

for i = 1:length(norad_id)

mount.SlewToCoordinatesAsync(tra_hrs(i),tdec_deg(i));

% wait until 2 minuts before pass
pause(datenum(t_pass(i,:))*d2s-now*d2s-2*60)

mount.SlewToCoordinatesAsync(tra_hrs(i),tdec_deg(i));
cam.LiveStart
% start imaging 30 sec before center prediction
pause(datenum(t_pass(i,:))*d2s-now*d2s-30)

% take images for 1 minute of 4 second exposures
for j = 1:15

cam.MemorySnapImage

% time at end of image exposure
dt = datestr(now,'yy_mm_dd_HH_MM_SS_FFF');

%bitmap file, can do others, tiff, jpeg, etc
dt = [pwd '\collected_data\' num2str(norad_id(i)) '_' dt '.bmp'];

cam.MemorySaveImage(dt)
end

cam.LiveStop

end

done = 1;

end
```

Appendix C

Operational Procedures

Surveillance Operation Procedures

As of Spring 2015, there is no network connection at the observatory so steps 1 through 3 must be completed with a network connection, before going to the observatory. You must be certified by the Virginia Tech Astronomy Club to operate the telescope and obtain the observatory key from Robeson Hall, Room 120.

1. Make sure you have the "telescope control" folder, set it to be the current MATLAB directory, and open the surveillance operation code "surveillance_operation_script.m".

2. Open the user pass input function: "nightly_tasking_user_input.m" and input the information on the satellites you wish to observe for the operational period, following the documentation in the function.

3. Run the first section of "surveillance_operation_script.m" by placing the cursor in the section and pressing ctrl-enter.

4. Run the second section of "surveillance_operation_script.m". If you get a timeout error from the EOP or TLE lookup functions, run the section again until you do not get an error.

5. Obtain the observatory key by checking it out. Reference the Observatory and Telescope Initialization Procedures, page 119. Follow 1. through 11. (excluding 2. and 3.).

6. Take off the camera lens protector, mount the camera on top of the telescope, and connect the camera to the control computer, using the second from the front USB port on the left side.

7. Connect the mount hand controller (RS-232 on bottom of hand controller) to the control computer, using the USB port closest to the front of the laptop on the left side.

8. Run the third section of "surveillance_operation_script.m". If either the mount or the camera will not connect, wait a few moments and run the section again. This section will open a live image view for camera focusing.

9. Focus the camera by rotating the front of the camera lens. View a bright star, using the hand control to point the mount and rotating the dome as necessary. Focus the camera to get the star as close to a point in the image as possible, as you get into better focus, less bright stars may become visible in the image. If you cannot view any stars, make sure you have rotated the dome, uncomment and run line 94 and tune the camera parameters as mentioned in the code.

10. When you are ready to begin mount operation (i.e. the mount will move on its own), run section 4a of "surveillance_operation_script.m". Depending on the time of the passes you are interested in, MATLAB will be busy for a while – do not interrupt.

11. When the mount slews to a new position, rotated the dome to allow sky viewing.

Note, If automatic operation gets interrupted, you can task the mount for an individual

satellite from the user input by running section 4b of "surveillance_operation_script.m" and setting "sat" to the order number of the satellite in the tasking lineup you wish to task.

Note, images are saved in the "collected_data" subfolder of "telescope control" with the naming convention "norad_id_yy_mm_dd_hh_mm_sssss" where the time listed is the end of the exposure for that image. Images are taken for 1 minute surrounding the time of the minimum range (see thesis section or code for further explanation).

12. "done = 1" will show in the command window when operation is complete. Disconnect the mount and camera from the control computer and remove the camera from the telescope.

13. Reference the Observatory and Telescope Shutdown Procedures. Follow the procedure (excluding 7. through 9.).

Observatory and Mount Procedures

Prices Fork Observatory Users Guide (9/30/2014)

Startup Procedure

NEVER MOVE OR POINT THE TELESCOPE MANUALLY!
NEVER TOUCH OR CLEAN THE TELESCOPE OPTICS!

1. Plug in the dome slit switch. Open the slit. Unplug the dome slit switch. If the slit will not open, stop now. There is no point in trying to boot the telescope.
2. Take the dust caps off telescope (front and eyepiece end), and the finder scope. Place these on the desk.
3. Put the large Axion LX eyepiece into the telescope. Secure it in place with set screws.
4. On the power strip cable-wrapped to the side of the telescope's vertical mount pole, make sure the switch is on (lighted red).
5. Plug the telescope's DC power connecter into the telescope pier. (The DC power unit sits at the base of the equatorial telescope mount.)
6. Turn on the telescope using the switch on the telescope pier.
7. The hand controller should say "Wake up. Press Enter to wake from hibernate." Press Enter. (If this did not happen, see the Appendix.)
8. You will be shown the current time (on a 24-hour clock display). Check the displayed time against your cell phone. Either accept the current time as correct, by hitting the Enter key, or enter time of the next approaching minute and wait until your cell phone reaches that minute, then hit the Enter key.
9. You will be asked to Select One of the following: Daylight Saving or Standard Time. Move through this list using the Up and Down keys (the 6 and 9 keys). Hit Enter when the appropriate selection is displayed. Repeat for the appropriate Time Zone (Eastern USA, of course).
10. You will be shown the current date. Hit Enter to accept the displayed date, or change it and hit Enter.
11. The hand controller will display "CGE Pro Ready."
12. Proceed to observing objects! Procedures for automatically moving the telescope to an object include
    a. Selecting a list by hitting one of the keys (M for Messier, Cald for Caldwell, NGC for NGC objects, etc.), then entering a value, or using the Up/Down buttons to scroll through the list, then hitting Enter (watch out --- the telescope will slew on its own to the selected object!).
    b. Hitting the List key, using the Up/Down keys (6 or 9) to scroll through the list of lists, then hitting the Enter key to select a list, etc.
    c. Hitting the Tour key to make the telescope show you a tour of objects.
    d. In any case, if you ever want to go back a step, hit the Undo key.
    e. Rotate the dome as necessary.
    f. Use the arrow keys to center any objects in the Telrad, and/or finder, and/or eyepiece.

Shutdown Procedure

1. Hit the Menu key. Scroll Down (key 9) to Utilities and hit the Enter key. Scroll Down (key 9) to Hibernate. Press Enter.
2. Set the slew rate to its highest value, by hitting the Rate key and the 9 key.
3. Use the arrow keys to put the telescope in a safe, stable position with its counter weights at their lowest and the telescope pointing approximately toward the North Celestial Pole (Polaris). DO NOT try to automatically point the telescope at Polaris!
4. Press the Enter key.
5. Turn off the telescope using the switch on the telescope pier.
6. Disconnect the DC power connector from the Telescope pier.
7. Take out the eyepiece and return it to its box.
8. Replace the lens caps on the telescope and finder.
9. Turn off the Telrad.
10. Rotate the dome so the slit is above the stairs. Plug in the dome slit switch. Close the slit. Unplug dome slit switch.
11. Turn out all lights, lock the observatory door.
12. Close the observatory gate.
13. Close and lock the pasture gate on the way out.

Appendix (Warning! More complicated than normal usage. Use only if you lose power to the telescope.)

If you lose power or the telescope does not properly wakeup, you will need to realign the telescope before using it. This is a more involved process than waking it up from hibernation. If you don't want to try this, turn the telescope off, leave and lock the observatory, then contact Dr. Simonetti (jhs@vt.edu) so he can realign the telescope. If you want to try realigning the telescope, this Appendix outlines the procedure.

1. With the telescope power on, press the Menu key, scroll Down to Utilities, and press Enter. Scroll Down to Factory Setting, and press Enter. The hand controller will say you need to press the zero key to proceed. Press the zero key. Then turn the telescope off, wait ten seconds, and turn the telescope on.
2. Display will say CGE Pro Ready. Press Enter. Hand controller will respond with "Set switch position." Press Enter again.
3. Display will say "Select One, Custom Site." Press Enter to select Custom Site.
4. Enter correct longitude of observatory (see sign on inside wall of observatory dome). Press Enter. And press Enter when display says "West."
5. Enter correct latitude of observatory (see sign on inside wall of observatory dome). Press Enter. And press Enter when display says "North."
6. The hand controller will prompt your for the current time (24-hour clock). Use your cell phone time to input a time that is on the next minute ahead of the current time and hit the Enter key when the minute changes on your cell phone (try to be accurate, it matters).
7. Select either Daylight Saving Time or Standard Time using the Up and Down keys and hit Enter.
8. Select the appropriate time zone (Eastern USA), and hit Enter.
9. Enter the appropriate date and hit Enter.
10. Choose Two Star Alignment by hitting Enter when it is displayed. These stars will be in the west. It will suggest a first star (all the suggested stars are bright stars). Hit Enter to go with its suggestion. The telescope will slew (move) to the star. If that star is not observable (behind clouds, or a tree), press the Undo key. It will suggest another star. When you get an observable star, center it in the Telrad using the arrow keys on the hand controller, then press Enter. Finally center the star in the eyepiece, and press the Align key. Repeat this procedure with a second suggested star.
11. It will ask you to "Add calib stars?" Follow the same selection and centering procedure, but all the calib stars (calibration stars) should be in the east. It will only allow for a few calib stars (four at most).
12. The display will say "CGE Pro Ready." You are now ready to begin observing. Go to the last step in the Startup Procedure to begin observing.

Appendix D

Simplified General Perturbations 4 Algorithm

The SGP4 method can be summarized as follows: The Kozai mean orbital elements and other orbital parameters at epoch, SGP4 mean motion n_0, eccentricity e_0, inclination i_0, mean anomaly M_0, argument of perigee ω_0, right ascension of ascending node Ω_0, time rate change of the mean motion $\dot{n}_0$, and SGP4 drag coefficient B^* are given through TLE lookup, and $t - t_0$ is the time since TLE epoch.

First Brouwer defined mean motion, n_0'', and semi-major axis, a_0'', are calculated from the Kozai mean elements reported by the TLE:

$$a_1 = \left(\frac{\sqrt{GM}}{n_0} \right)^{2/3}$$

$$k_2 = \frac{1}{2} J_2 a_E^2$$

$$\delta_1 = \frac{3k_2(3\cos^2 i_0 - 1)}{2a_1^2(1 - e_0^2)^{\frac{3}{2}}}$$

$$a_0 = a1(1 - \tfrac{1}{3}\delta_1 - \delta_1^2 - \tfrac{134}{81}\delta_1^3)$$

$$\delta_0 = \frac{3k_2(3\cos^2 i_0 - 1)}{2a_0^2(1-e_0^2)^{\frac{3}{2}}}$$

$$n_0'' = \frac{n_0}{1+\delta_0}$$

$$a_0'' = \frac{a_0}{1-\delta_0} \tag{D.1}$$

Then constants can be calculated based on the mean epoch elements from the TLE:

$$\theta = \cos i_0$$

$$\xi = \frac{1}{a_0'' - s}$$

$$\beta_0 = \sqrt{1-e_0^2}$$

$$\eta = a_0'' e_0 \xi$$

$$C_2 = (q_0 - s)^4 \xi^4 n_0''(1-\eta^2)^{-\frac{7}{2}}[a_0''(1+\tfrac{3}{2}\eta^2 + 4e_0\eta + e_0\eta^3) +$$

$$\frac{3k_2\xi}{2(1-\eta^2)}(-\tfrac{1}{2} + \tfrac{3}{2}\theta^2)(8 + 24\eta^2 + 3\eta^4)]$$

$$C_1 = B^* C_2$$

$$A_{3,0} = -J_3 a_e^3$$

$$C_3 = \frac{(q_0 - s)^4 \xi^5 A_{3,0} n_0'' a_E \sin i_0}{k_2 e_0}$$

$$C_4 = 2n_0''(q_0 - s)^4 \xi^4 a_0'' \beta_0^2 (1-\eta^2)^{-\frac{7}{2}}([2\eta(1+e_0\eta) + \tfrac{1}{2}e_0 + \tfrac{1}{2}\eta^3] - \frac{2k_2\xi}{a_0''(1-\eta^2)}$$

$$[3(1-3\theta^2)(1 + \tfrac{3}{2}\eta^2 - 2e_0\eta - \tfrac{1}{2}e_0\eta^3) + \tfrac{3}{4}(1-\theta^2)(2\eta^2 - e_0\eta - e_0\eta^3)\cos 2\omega_0])$$

$$C_5 = 2(q_0 - s)^4 \xi^4 a_0'' \beta_0^2 (1-\eta^2)^{-\frac{7}{2}} [1 + \tfrac{11}{4}\eta(\eta + e_0) + e_0\eta^3]$$

$$D_2 = 4a_0'' \xi C_1^2$$

$$D_3 = \tfrac{4}{3} a_0'' \xi^2 (16a_0'' + s) C_1^3$$

$$D_4 = \tfrac{2}{3} a_0'' \xi^3 (221a_0'' + 31s) C_1^4 \tag{D.2}$$

Secular effects of atmospheric drag and gravitation are calculatede as:

$$M_{DF} = M_0 + \left[1 + \frac{3k_2(-1+3\theta^2)}{2a_0''^2\beta_0^3} + \frac{3k_2^2(13 - 78\theta^2 + 137\theta^4)}{16a_0''^4\beta_0^7}\right] n_0''(t - t_0)$$

$$k_4 = -\tfrac{3}{8} J_4 a_E^4$$

$$\omega_{DF} = \omega_0 + n_0''(t - t_0)\left[- \frac{3k_2(1-5\theta^2)}{2a_0''^2\beta_0^4} + \frac{3k_2^2(7 - 114\theta^2 + 395\theta^4)}{16a_0''^4\beta_0^8} + \frac{5k_4(3 - 36\theta^2 + 49\theta^4)}{4a_0''^4\beta_0^8}\right]$$

$$\Omega_{DF} = \Omega_0 + \left[-\frac{3k_2\theta}{a_0''^2\beta_0^4} + \frac{3k_2^2(4\theta - 19\theta^3)}{2a_0''^4\beta_0^8} + \frac{5k_4\theta(3 - 7\theta^2)}{2a_0''^4\beta_0^8}\right] n_0''(t - t_0)$$

$$\delta\omega = B^* C_3 \cos\omega_0(t - t_0)$$

$$\delta M = -\tfrac{2}{3}(q_0 - s)^4 B^* \xi^4 \frac{a_E}{e_0\eta}[(1 + \eta\cos M_{DF})^3 - (1 + \eta\cos M_0)^3]$$

$$M_p = M_{DF} + \delta\omega + \delta M$$

$$\omega = \omega_{DF} - \delta\omega - \delta M$$

$$\Omega = \Omega_{DF} - \frac{21 n_0'' k_2 \theta}{2 a_0''^2 \beta_0^2} C_1 (t - t_0)^2$$

$$e = e_0 - B^* C_4 (t - t_0) - B^* C_5 (\sin M_p - \sin M_0)$$

$$a = a_0'' [1 - C_1 (t - t_0) - D_2 (t - t_0)^2 - D_3 (t - t_0)^3 - D_4 (t - t_0)^4]^2$$

$$IL = M_p + \omega + \Omega + n_0'' [\tfrac{3}{2} C_1 (t - t_0)^2 + (D_2 + 2C_1^2)(t - t_0)^3 + \tfrac{1}{4}(3D_3 + 12 C_1 D_2 + 10 C_1^3)(t - t_0)^4 + \tfrac{1}{5}(3D_4 + 12 C_1 D_3 + 6 D_2^2 + 30 C_1^2 D_2 + 15 C_1^4)(t - t_0)^5]$$

$$\beta = \sqrt{1 - e^2}$$

$$n = \frac{GM}{a^{\frac{3}{2}}} \tag{D.3}$$

Long periodics are calculated as:

$$a_{xN} = e \cos \omega$$

$$IL_L = \frac{A_{3,0} \sin i_0}{8 k_2 a \beta^2} (e \cos \omega) \frac{3 + 5\theta}{1 + \theta}$$

$$a_{yNL} = \frac{A_{3,0} \sin i_0}{4 k_2 a \beta^2}$$

$$IL_T = IL + IL_L$$

$$a_{yN} = e \sin \omega + a_{yNL}$$

To calculate short periodics Kepler's equation must be solved for $E + \omega$ using:

$$U = IL_T - \Omega$$

$$(E+\omega)_i = U$$

$$(E+\omega)_{i+1} = (E+\omega)_i + \Delta(E+\omega)_i$$

$$\Delta(E+\omega)_i = \frac{U - a_{yN}\cos(E+\omega)_i + a_{xN}\sin(E+\omega)_i - (E+\omega)_i}{-a_{yN}\sin(E+\omega)_i - a_{xN}\cos(E+\omega)_i + 1} \tag{D.4}$$

Short periodics can then be calculated as:

$$e\cos E = a_{xN}\cos(E+\omega) + a_{yN}\sin(E+\omega)$$

$$e\sin E = a_{xN}\sin(E+\omega) - a_{yN}\cos(E+\omega)$$

$$e_L = (a_{xN}^2 + a_{yN}^2)^{\frac{1}{2}}$$

$$p_L = a(1-e_L^2)$$

$$r = a(1-e\cos E)$$

$$\dot{r} = \frac{\sqrt{GMa}}{r} e\sin E$$

$$r\dot{f} = \frac{\sqrt{GMp_L}}{r}$$

$$\cos u = \frac{a}{r}\left[\cos(E+\omega) - a_{xN} + \frac{a_{yN}e\sin E}{1+\sqrt{1-e_L^2}}\right]$$

$$\sin u = \frac{a}{r}\left[\sin(E+\omega) - a_{yN} - \frac{a_{xN}e\sin E}{1+\sqrt{1-e_L^2}}\right]$$

$$\Delta r = \frac{k_2}{2p_L}(1-\theta^2)\cos 2u$$

$$\Delta u = -\frac{k_2}{4p_L^2}(7\theta^2 - 1)\sin 2u$$

$$\Delta\Omega = \frac{3k_2\theta}{2p_L^2}\sin 2u$$

$$\Delta i = \frac{3k_2\theta}{2p_L^2}\sin i_0 \cos 2u$$

$$\Delta\dot{r} = -\frac{k_2 n}{p_L}(1-\theta^2)\sin 2u$$

$$\Delta r\dot{f} = \frac{k_2 n}{p_L}[(1-\theta^2)\cos 2u - \tfrac{3}{2}(1-3\theta^2)] \tag{D.5}$$

Short periodics are added to find the osculating, or instantaneous, elements:

$$r_k = r[1 - \tfrac{3}{2}k_2\frac{\sqrt{1-e_L^2}}{p_L^2}(3\theta^2 - 1)] + \Delta r$$

$$u_k = u + \Delta u$$

$$\Omega_k = \Omega + \Delta\Omega$$

$$i_k = i_0 + \Delta i$$

$$\dot{r}_k = \dot{r} + \Delta\dot{r}$$

$$r\dot{f}_k = r\dot{f} + \Delta r\dot{f} \tag{D.6}$$