



**DEVELOPMENT OF A MODULARIZED SOFTWARE ARCHITECTURE TO
ENHANCE SSA WITH COTS TELESCOPES**

THESIS

Julian P. McCafferty, 2d Lt, USAF

AFIT-ENY-MS-16-M-227

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-16-M-227

DEVELOPMENT OF A MODULARIZED SOFTWARE ARCHITECTURE TO
ENHANCE SSA WITH COTS TELESCOPES

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Astronautical Engineering

Julian P. McCafferty, BS Aerospace Engineering

2d Lt, USAF

March 2016

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENY-16-M-227

DEVELOPMENT OF A MODULARIZED SOFTWARE ARCHITECTURE TO
ENHANCE SSA WITH COTS TELESCOPES

Julian P. McCafferty, BS Aerospace Engineering

2d Lt, USAF

Committee Membership:

Dr. Richard Cobb
Chair

Dr. Bradley Ayres
Member

Capt Robert Bettinger, PhD
Member

Abstract

As the catalog of Earth orbiting objects continues to grow exponentially, so too does the necessity for Space Situational Awareness (SSA). The U.S. Air Force has taken the lead in providing the world with near real-time SSA using its Space Surveillance Network (SSN) consisting of terrestrial and on-orbit optical and radar sensors. Previous work at AFIT has explored augmenting the SSN by demonstrating detection and tracking of orbiting objects using Commercial-Off-The-Shelf (COTS) telescopes and Air Force generated Two-Line Element Sets (TLE). Although this capability has been proven, achieving practical application of this technology requires careful software design and hardware testing. This research explores the process of developing and reengineering code into a modularized, hierarchical component architecture designed for the end user while enabling developers to continue to modify the software for future applications.

Three graphical user interfaces (GUI) are compiled into standalone executable programs using MATLAB for propagating orbits, orbit mapping, and performing observations. Previous AFIT in-house MATLAB code is further developed to be hardware agnostic and continue to operate on future operating systems. Finally, open-loop optical tracking of low Earth orbit (LEO) satellites is demonstrated using a Meade LX200GPS telescope and Alt-Az mount with 100% of targets captured in the Orion 80mm spotting scope. The ultimate result of this work is a set of modular tools available to students and researchers for orbit propagation, mission planning, and satellite tracking which can be further developed to realize a cost saving technology to meet the Air Force's growing demand for SSA.

Acknowledgments

I would first like to acknowledge my advisor Dr. Cobb for tolerating an Aeronautical Engineer learning how to program a telescope to track satellites. After 18 months, I am still amazed how any of that happened. Only by giving me the freedom and guidance during my research was I able to become acquainted with the fascinating world of software, telescopes, and space tracking. I would also like to acknowledge the technical and hands on support of David Shultz. His knowledge and presence setting up the telescopes, answering endless questions, and preparing for a night of observations made testing successful and enjoyable. Finally, I should acknowledge my student peers for making the late nights bearable and the long hours entertaining. AFIT is not a solitary accomplishment.

Julian P. McCafferty

Table of Contents

	Page
Abstract	iv
Table of Contents	vi
List of Figures	ix
List of Tables	xii
I. Introduction	1
Motivation	1
Background.....	5
Problem Statement.....	6
Research Question	6
Research Objectives	6
Research Approach.....	7
Assumption/Limitations	9
Implications	9
Summary.....	10
II. Background	11
Chapter Overview.....	11
Relevant Research	11
Relevant AFIT Research	15
Software Design	19
Modularity	24
Telescope Hardware	27
Summary.....	29

III. Methodology	30
Chapter Overview	30
Equipment.....	30
Software Architecture.....	34
Modularity	49
Testing	51
Summary.....	54
IV. Results.....	55
Chapter Overview	55
Test Results	55
Hardware Independence	65
Summary.....	68
V. Conclusions.....	69
Overview	69
Conclusions of Test Results (Objective 1)	70
Software Architecture (Objective 2)	71
Compile Tracking Utilities (Objective 3).....	72
Mission Planning and Orbit Mapping Tool (Objective 4).....	73
Integrate Closed Loop Tracking Functionality (Objective 5)	76
Demonstrate Tracking on an Alt-Az and Equatorial Mount (Objective 6)	76
Significance of Research	76
Future Work.....	77
Lessons Learned	79
Summary.....	81

Appendix A – TeleTrak Quickstart Guide.....	82
Appendix B – TeleCalc Overview	85
Appendix C – MATLAB Deploy Tool.....	90
Appendix D – Architecture and Design Documents.....	92
TeleCalc Architecture Design	92
TeleMap Architecture Design	95
TeleTrak_v8 Architecture Design	96
Bibliography	97

List of Figures

	Page
Figure 1: Satellite Catalog Growth [1].....	1
Figure 2: US Space Surveillance Network (SSN) [5].....	3
Figure 3: AFIT Tracking Station History Through 2013 [16].....	16
Figure 4: Software Development Process [20]	20
Figure 5: SciBox Hierarchical Component Design [15].....	21
Figure 6: Two-Line Element Set Definition [21]	22
Figure 7: Modular Software Design Concept [22]	25
Figure 8: Compound Telescope Diagram [23]	27
Figure 9: Refractor Telescope Diagram [23].....	28
Figure 10: Types of Telescope Mounts [23].....	28
Figure 11: Lab Telescope Configuration	31
Figure 12: Orion ShortTube 80mm Refractor Telescope Optical Assembly [24].....	31
Figure 13: Astrovid StellaCam III [25].....	32
Figure 14: Epiphan VGA2USB Frame Grabber [28]	34
Figure 15: TeleTrak Hierarchical Component Design.....	38
Figure 16: Earth Centered Inertial Frame [28]	40
Figure 17: Earth Centered Earth Fixed Frame.....	40
Figure 18: South-East-Zenith Frame [28].....	40
Figure 19: SEZ Reference Frame Relation to Az-El [17].....	40
Figure 20: TeleCalc Interface	44
Figure 21: TeleMap Interface	46

Figure 22: TeleMap Ground Site Tab.....	46
Figure 23: TeleMap Globe Tab.....	47
Figure 24: TeleTrak_v8 Interface	48
Figure 25: TeleCalc Dependency Diagram.....	49
Figure 26: TeleTrak Modularity	50
Figure 27: AFIT Rooftop Telescope Configuration	53
Figure 28: AFIT Ground Station.....	53
Figure 29: Difference between TeleCalc and STK ISS Position for Single Orbit.....	56
Figure 30: ISS Orbit in TeleMap STK vs TeleCalc.....	56
Figure 31: Difference between TeleCalc and STK ISS Position Vector Magnitude.....	57
Figure 32: Distance between ISS Position According to STK and TeleCalc	57
Figure 33: Difference between TeleCalc and STK ISS Elevation Angle from AFIT	59
Figure 34: Difference between TeleCalc and STK ISS Azimuth Angle from AFIT.....	59
Figure 35: ISS Pass Start 11:58:42 UTC	60
Figure 36: ISS Pass End 12:03:36 UTC	60
Figure 37: Ground Site FOV.....	60
Figure 38: ISS Ground Track.....	60
Figure 39: TeleTrak_v8 during ISS Pass, 6 Jan 2016.....	61
Figure 40: Orion Spotting Scope and Meade Main Optic Field of View	62
Figure 41: COSMOS 1544.....	63
Figure 42: SL-16 R/B.....	63
Figure 43: COSMOS 2151.....	63
Figure 44: CZ4B R/B.....	63

Figure 45: RESURS-DK 1	64
Figure 46: SL-3 R/B.....	64
Figure 47: ISS Low Elevation.....	64
Figure 48: ISS High Elevation.....	64
Figure 49: Screen Capture Technique for Optical Feedback.....	67
Figure 50: Operational GPS Constellation in ECEF Frame	74
Figure 51: Operational GPS Constellation in ECI Frame.....	74
Figure 52: Full TLE Catalog Plots.....	75
Figure 53: Full TLE Catalog Plot, GEO Regime.....	75
Figure 54: Full TLE Catalog Ground Track	76

List of Tables

Table 1: Telescope - Camera Configuration Properties.....	33
Table 2: Reference Data File Descriptions	41
Table 3: TeleCalc Output Definitions.....	42
Table 4: Success Rate of Observations	64
Table 5: Research Objectives Met	69

DEVELOPMENT OF A MODULARIZED SOFTWARE ARCHITECTURE TO ENHANCE SSA WITH COTS TELESCOPES

I. Introduction

Motivation

With the increasing reliance on space assets, a gap exists between the demand for space situational awareness (SSA) and the current capabilities and available resources for acquiring data on resident space objects (RSOs). Resident space objects include earth orbiting satellites, space debris, and near earth asteroids. In the 2011 National Security Space Strategy, space is described as an increasingly congested, contested, and competitive environment [1]. Space is congested because of the satellite catalog growth, shown in Figure 1, and increasing number of space faring nations. This issue became self-evident in 2009 when the Russian Cosmos satellite collided with the U.S. commercial Iridium satellite, creating approximately 1,500 pieces of trackable space debris, adding to the more than 3,000 pieces of debris created by the 2007 Chinese anti-satellite test [1].

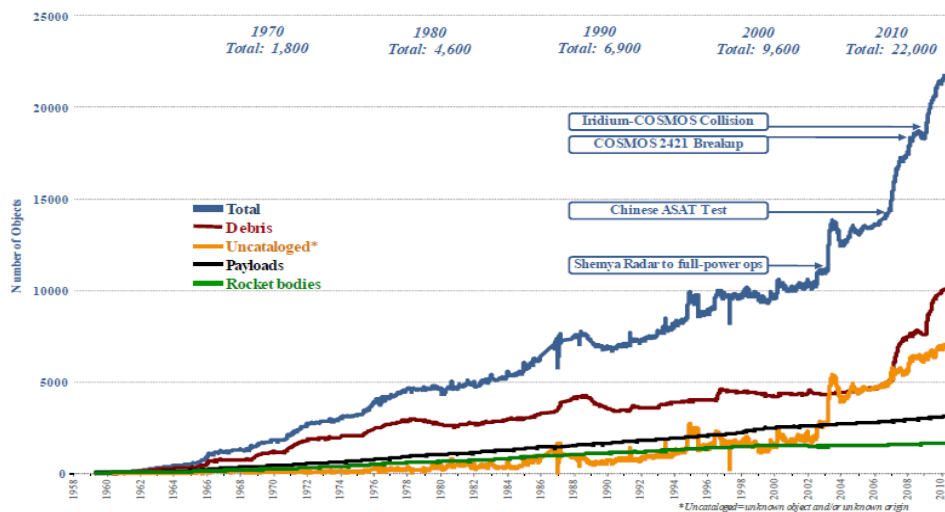


Figure 1: Satellite Catalog Growth [1]

Space is contested because of the increasing number of man-made threats intended to deny and disrupt space systems. Some are known, but many have yet to be identified, further requiring SSA and satellite characterization capabilities. Space is competitive because the industry is closely tied to the world economy and dependent on finite resources and orbital slots. While the barrier to entry for space is high, the benefits demonstrated by space faring nations are desirable and lead to a competitive market.

While the National Space Strategy recognizes that space is a congested, contested, and competitive environment, the U.S. also recognizes the unprecedented advantages in national decision-making, military operations, and homeland security that space capabilities provide, thereby motivating the need for space assurance and situational awareness. Naturally the focus for Air Force Space Command has been on SSA, but the need and complexity of SSA has been escalating while the dedicated resources have not. Commander of Air Force Space Command, Gen William Shelton said in a 2014 speech, “I can’t think of a single military operation that doesn’t rely on space” [2]. While the U.S. recognizes the critical nature of SSA, current SSA dedicated resources may not be able to meet the vital demand in the future. In September of 2013, the Air Force Space Surveillance System (also known as the Air Force Space Fence) was shut down [3]. This network of ground sites detected satellites up to 24,000 km by reflecting VHF radio signals and was responsible for approximately 40% of all Air Force Space Surveillance observations [3]. Scheduled to begin operations in 2017, the contract for a new Space Fence using an S-band radar system was awarded to Lockheed Martin consisting of a site in Kwajalein Atoll and an optional site in Western Australia

[4]. Already this program has slipped to initial operability in 2019 and full operability in 2022 [4]. While the new Space Fence boasts much greater effectiveness, radar sites at Eglin AFB and Cavalier Air Force Station have had to pick up the slack during the interim period. Satellite orbital estimation and tracking through radar, optical measurement, and catalog maintenance are all critical to the total SSA mission. Figure 2 below shows the currently operational Air Force ground sites and space assets dedicated to SSA.

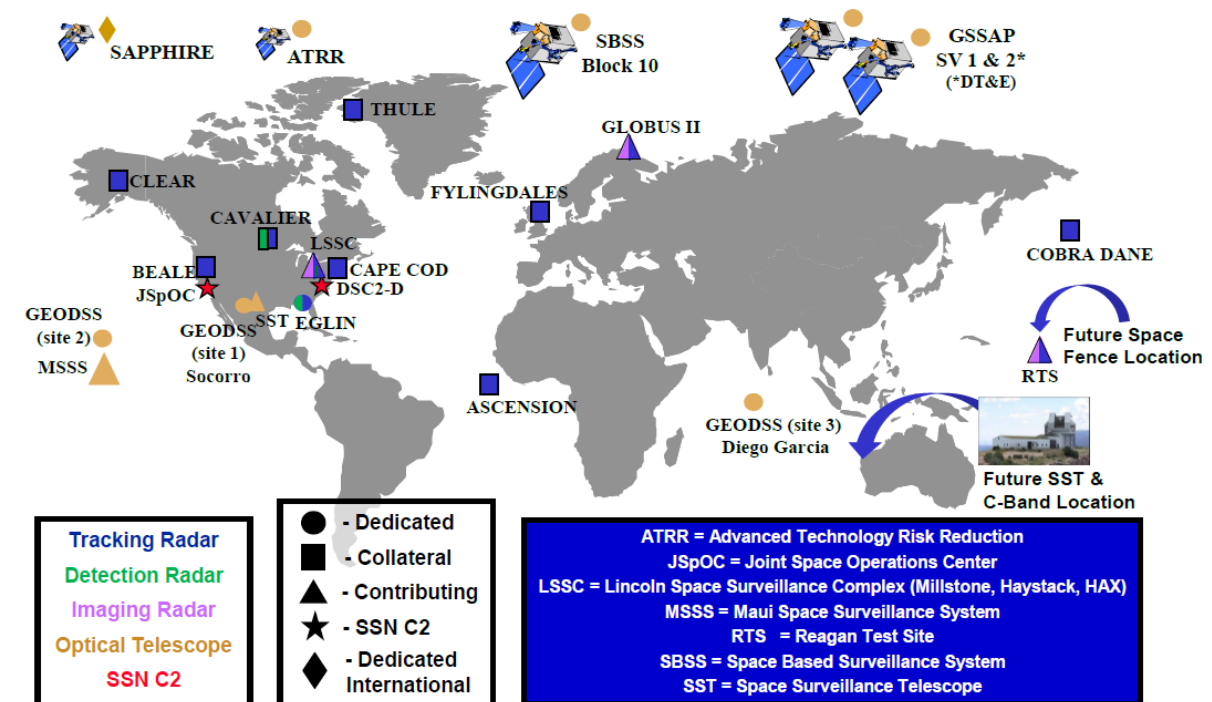


Figure 2: US Space Surveillance Network (SSN) [5]

Figure 2 illustrates the whole system architecture required for the SSA mission, and how very few sites are dedicated to optical tracking and characterization. As of 1 July 2015, the US Space Surveillance Network has cataloged 16,925 objects currently orbiting earth, 3,917 of which are payloads, and this orbital catalog has been increasing exponentially [6]. The demand for spacecraft characterization has become greater with

the advent of unknown adversarial capabilities and anomalous activity in the space domain. In addition to the need to determine the health and status of our own space assets, a need exists for electro-optical tracking of RSOs with little to no knowledge of an impending threat. As the quantity of orbiting objects and frequency of potentially threatening space activity grows, current Air Force operated ground sites will be challenged to meet the demand for SSA; however, the deployment of low cost automated tracking and characterization ground stations are an attainable solution to this problem.

Previous research at the Air Force Institute of Technology (AFIT) has recognized and addressed the demand for low cost, flexible, automated ground-based optical tracking, detection, and characterization of RSOs for the enhancement of SSA. The AFIT TeleTrak system was founded as a research platform for this purpose. This research is motivated by the loss in TeleTrak capability due to software constraints and lack of an established flexible, reusable software design structure. One of the greatest challenges designers face when delivering software products is considering the interface between the user and the tool. The most powerful tools have little value if the interface is not intuitive and understandable to the user. Furthermore, a system without flexibility in terms of its hardware and operating environment is unreliable and a huge risk to its applied mission. This represents the challenges presented in TeleTrak because it has been built and programmed around specific hardware with outdated software and operating systems. The objective of TeleTrak is to demonstrate low cost methods for space resiliency and assurance with commercial hardware, but the system itself must also exemplify resiliency and reliability. This is the unique challenge to using

commercial rather than dedicated hardware and software because it is subject to periodic updates based on the commercial sector. Addressing this challenge with TeleTrak is critical to the case for saving time and money with COTS components to fill a variety of SSA needs.

Background

The Air Force's focus on SSA has motivated years of research at AFIT and the establishment of a satellite tracking ground station affectionately known as "TeleTrak". This system consists of commercial telescopes and equipment capable of tracking and recording LEO and geostationary orbit (GEO) satellites by pointing the telescope based on propagated orbits using Two-Line Element sets (TLEs) published every 12 hours by the North American Aerospace Defense Command (NORAD) on Space-Track [7]. The purpose of this tracking station is to provide a hands-on educational tool for demonstrations and research in the development and operation of a satellite tracking ground site.

As of this writing, the AFIT TeleTrak tools have previously demonstrated the ability to detect and acquire a known satellite from a propagated TLE taken from space-track.org, track the satellite's position using closed-loop optical feedback from the spotting scope of a Meade LX200GPS telescope, and then record video through the main optic. The most recent results demonstrated tracking with a satellite target spending an average of 93.0% of the time within a 0.05° field of view (FOV) in addition to autonomous tracking [8]. After 2012, research switched from LEO to GEO observations, thus no longer requiring the TeleTrak software. Since then, TeleTrak

operability has been lost due to commercial software and hardware updates. Expanding the TeleTrak research objectives to both LEO and GEO tracking as well as incorporating updated in-house TeleTrak software for previous Alt-Az mounts and new Equatorial mounts motivates this work. Furthermore, designing a fully automated capability for TeleTrak for remote operations will enable greater data collection for analysis of candidate algorithms.

Problem Statement

The AFIT TeleTrak network has previously demonstrated tracking with COTS telescopes, but capability has been lost due to specific hardware and software constraints that are not well understood. Furthermore, a common organized repository for TeleTrak code has not been established and sharing or replicating telescope and tracking performance has not been possible. Updating the in-house TeleTrak software for hardware flexibility and modern operating systems is necessary for the development of TeleTrak for real-world SSA data acquisition.

Research Question

How to best reengineer existing code and develop a set of modularized software tools for orbit propagating, mission planning, and satellite tracking with COTS telescopes that are hardware agnostic and adaptable to future operating systems.

Research Objectives

Primary Objectives:

1. Regain previous open-loop tracking functionality of maintaining a target within the spotting scope FOV for the duration of a pass.

2. Establish a modular software architecture.
3. Compile tracking utilities into standalone executable programs for Windows and MAC operating systems.
4. Develop an orbit mapping and observation mission planning tool.

Secondary Objectives:

5. Regain previous closed-loop tracking functionality of greater than 90% target time spent within a 0.05° FOV.
6. Demonstrate desired tracking functionality on an Alt-Az and Equatorial telescope mount.

Research Approach

The approach for this research is to first reorganize the current AFIT tracking MATLAB code into a new set of repositories organized by function. All of the legacy code was stored into folders by the student that used, it rather than by purpose or version control. Reorganizing entails identifying the most recent and previously operational sets of code and decomposing the MATLAB protocols and processes into their basic functions to understand their purpose and rationale. A step-by-step organizational document will then be constructed illustrating the program routines by the order and process by which they are called.

Once the most recent code is identified and analyzed, the code will be updated and rewritten to operate in MATLAB 2015a on Windows 7 OS with a Meade LX200GPS telescope. Regaining previous functionality allows the hardware and software to be tested and verified for the next steps.

With updated and operational tracking software, the code can be further developed for flexibility and ease of use. Flexibility will be accomplished by

implementing hardware agnostic software such that hardware interfaces can be updated by switching out a set of command translations rather than rewriting code or changing user interfaces. Ease of use includes user interfaces for data input and telescope operation and automating all processes that do not require user input. Furthermore, designing graphical user interfaces that are intuitive, robust, and provide the user with feedback minimizes frustration and potential for serious data errors.

Achieving modularity is a key aspect to the usability and customizability of the software system. Modularity can be achieved on multiple levels beginning with dividing the system into the following three mission areas: Orbit Propagation, Mapping and Planning, and Telescope Commanding. Within these mission areas, individual protocols can be divided out to their own functions to enable a developer to modify specific algorithms and customize user interfaces to achieve a desired output without impacting the overall software architecture. This also empowers users to identify a particular function or protocol and apply it to a different application or replace it with a new function, making the tools separable from the total system and individually available to users outside of TeleTrak.

The final step includes testing and validation of the software tools and making them available to future students and researchers with a variety of tracking and research objectives. In addition to sharing source code functions, availability can also be achieved by compiling the software tools into standalone executable programs that can run from different computers independent of the original software language program; in this case, MATLAB. The hardware and software tools used in this research are discussed in Chapter III.

Assumption/Limitations

This research is limited to the hardware and operating systems available at AFIT. It is assumed that other students and researchers interested in the TeleTrak software tools will have available the same or similar hardware and operating systems such that the tools can be tailored to their application. Furthermore, it is assumed that NORAD will continue to provide TLEs and other input data TeleTrak relies upon for the foreseeable future. In the likely event that the TLE standard and method of acquiring orbit data changes, TeleTrak will need to be adaptable to these updates. Previous theses and research have addressed the development of the original TeleTrak hardware and software tools, as well as analysis of their accuracy and reliability. The scope of this work focuses on reengineering the TeleTrak software tools and regaining previously demonstrated capability. Verification and validation of the tools in this research will be limited to satellite tracking of LEO satellites using a Meade mount; however, the hardware and software is capable of many more types of tracking and applications.

Implications

The impact of this research is that the AFIT TeleTrak hardware and software tools which include orbit propagation, mission planning, and satellite tracking will be made functional, flexible, and available to future students, researchers, and space operators. The TeleTrak software architecture will be organized into modularized, customizable functions and standalone executable programs which will be deliverable to students and researchers with a variety of SSA research objectives. Furthermore,

TeleTrak will be hardware agnostic and resilient to commercial updates so that it can remain functional in the future. The organization and availability of these hardware and software tools will enable TeleTrak to continue development to eventually realize a cost saving technology to meet the Air Force's growing demand for SSA.

Summary

Chapter I identifies a significant need in the Air Force for SSA. Previous research at AFIT has demonstrated the capability of supporting this mission with real-time optical satellite tracking using COTS telescopes; however, lack of fundamental software architecture design and documentation significantly limits software dependent capabilities. This research addresses the importance of the software design process and explores methods in reengineering and reorganizing code into a set of easy to use, flexible, and robust tools.

Chapter II presents a review of related literature, tools, and research in the field of satellite tracking with commercial telescopes as well as what can be learned from them and how they apply to this project. Chapter III details the methodology and research approach to modularizing and improving upon TeleTrak. Chapter IV presents the results and analysis and its implications. Finally, Chapter V summarizes the findings and discusses the impact on the future of TeleTrak and its application to enhancing Air Force SSA.

II. Background

Chapter Overview

The purpose of this chapter is to provide additional background and an evaluation of relevant research on the topic of satellite tracking with commercial telescopes. A review of relevant research on the topic of satellite tracking with COTS telescopes discussed in this chapter appears primarily from the Air Force Research Laboratory (AFRL), Advanced Maui Optical and Space Surveillance Technologies (AMOS) Conference publications, American Institute of Aeronautics and Astronautics (AIAA) publications, and the commercial sector. This research builds upon several years of TeleTrak development at AFIT outlined in this chapter. Finally, a background in software development is presented with the implications on this research.

Relevant Research

AFRL has previously demonstrated using small aperture COTS telescopes, known as Raven-class telescopes, for research and aiding SSA. A Raven-class system is not defined by hardware or software, but by the objective to develop a cost-effective system consisting of COTS based hardware and software [9]. Examples include supporting the Space Surveillance Network by optimizing for operational deep-space metrics, and Ravens for R&D by optimizing for photometry. The Raven-class telescope concept perfectly represents the objectives of TeleTrak and provides an example of a successfully deployed asset. In 2001 a 0.4 meter Raven-class telescope was deployed at the Maui Space Surveillance System (MSSS) contributing to the SSN, and AFRL has demonstrated successful tracking of LEO satellites with similar COTS equipment from

multiple remote sites [9]. AFRL has also conducted multiple studies on satellite attitude determination from Raven-class telescopes with great success. An example is the High Accuracy Network Determination System (HANDS) which utilizes Raven-class sensors to provide observations with less than an arcsecond of error [10]. HANDS was implemented to greatly augment orbit estimations by adding highly accurate angular observations to ranging observations compared to range observations alone.

Efforts have also been made to provide COTS telescopes to amateur astronomers and the wider astronomy community with the agreement that a portion of the observing time is dedicated to SSA [11]. This concept is borrowed from programs such as DARPA's Grand Challenge and the National Weather Service's program to provide government equipment to the community for a mutually beneficial relationship. Studies have shown this to be an extremely cost effective solution dramatically increasing SSA observations and reducing ground-based optical SSA operational costs by an order of magnitude [11]. This also provides greater motivation for the simplification and automation of the software architecture of a COTS ground station to enable this type of relationship with a community of amateur astronomers.

An example of a commercially available product for space surveillance is ExoAnalytic Space Operations Center (ESpOC™) by ExoAnalytic. ESpOC™ is an advanced software suite that performs real-time space object detection, tracking, and characterization [12]. Their suite provides distributed command and control of a network of telescopes and performs real-time processing of the focal plane data. Similar to TeleTrak, ExoAnalytic utilizes COTS telescopes ranging from 16'' to 4.5'' in aperture and ATIK 383L+ and ATIK 314L+ cameras [13]. While ESpOC™ is a

software suite and most of ExoAnalytic products are software based, they offer tasking of their network of small telescopes to commercial satellite operators and are additionally interested in outfitting Space Command with a network of ground-sites [13]. Many of the software tools made available by ExoAnalytic are dedicated to image processing, while different types of tracking and telescope tasking will be focused on in this research. Still, the modular architecture of the ESpOC™ software suite is a good example of making specific software tools available to the user. Astronomy Online lists a host of other commercially available software that perform functions ranging from telescope commanding and celestial target planning to CCD control and imaging [14]. Rarely is a single COTS software suite a “one stop shop” for the desired application and each program typically excels in a single aspect while lacking in another. Furthermore, commercial software does not provide the desired customizability for AFIT TeleTrak and inhibits the TeleTrak educational objective of learning through applied research; therefore, TeleTrak uses software developed in-house at AFIT.

An example of a software program in the astronomical engineering community that has successfully demonstrated principles of modularity, reusability, and rapid development is SciBox, a software library for rapid development of science operation simulation, planning, and command tools [15]. This set of software tools developed at the Johns Hopkins University Applied Physics Laboratory (APL) was not developed for any application related to telescopes; however, many lessons can be learned from its software design philosophy and proven track record. SciBox has aided in the design and decision making of multiple space science missions using dissimilar spacecraft and instrumentation operating in different environments. Developed over approximately 15

years, the objective of SciBox became developing a high-fidelity Operation Simulation Tool (OST) that could be rapidly customized and built upon for a diverse set of future science missions. This alleviates a significant challenge in the space industry which is the substantial time and money required to develop specific simulation software tools to generate requirements analysis during the early stages of mission design. This motivated the design of SciBox as a software “platform” that could be rapidly customized to a new mission by building off of a previous foundation. The SciBox concept holds a lot of value to the TeleTrak mission. Incorporating a similar software architecture that can be built upon from the highest level possible minimizes redundant work and creates a system that is customizable and flexible to a variety of applications within SSA. Choo, one of the creators of SciBox at APL, achieved the objectives of modularity, reusability, and rapid development by implementing a Hierarchical Component Design (HCD). Choo shared insights with the author into how building SciBox from the ground up using the HCD exhibits qualities of reusability and modularity and enables rapid development from a proven platform. For example, APL was tasked with designing a simulation tool to aid in the mission planning process for the MESSENGER spacecraft whose mission was to map the surface of Venus [15]. A flyby analyzer was rapidly assembled by selecting existing SciBox modules and adapting the top layer of the architecture to the environment and instruments of the MESSENGER spacecraft, saving considerable time and money in the mission planning process. A common mistake observed in software development is designing modeling tools from the ground up for a specific mission. The disadvantages to developing dedicated software are longer development times and increased risk due to untested and

unverified tools impacting key mission configuration decisions. The process by which SciBox was developed using the HCD influenced much of the design strategy and organization of TeleTrak discussed in Chapter III.

Relevant AFIT Research

Several years of research have been conducted utilizing the AFIT ground Station to study and demonstrate the value of COTS telescopes and other commercially or readily available tools for SSA. A visual review of previous research is presented in Figure 3. Not included in this figure is research by Moomey from 2014-2015. Previous work on TeleTrak most relevant to this research includes Schmunk in 2008 on Initial Orbit Determination (IOD) of LEO satellites, Briggs in 2011 on satellite detection and real-time IOD, Gresham in 2012 on closed-loop optical control tracking, and Moomey in 2015 on GEO SSA with small telescopes. Other theses that have contributed to SSA include Satellite Position Attained by RF-Keyed Tracking (SPARK) and TeleTrakNet. The SPARK project has supported multiple theses investigating the potential of detecting artificial satellites using reflected RF signals from the Air Force Space Surveillance System. The goal of SPARK is to use detection information from RF signals to inform and cue TeleTrak optical tracking for improved SSA. The TeleTrakNet project includes research into the feasibility and process of deploying a remotely operated telescope network. While other theses have utilized TeleTrak in some way from photometry to image processing, only the theses that contributed to the development of the TeleTrak software will be discussed in detail.

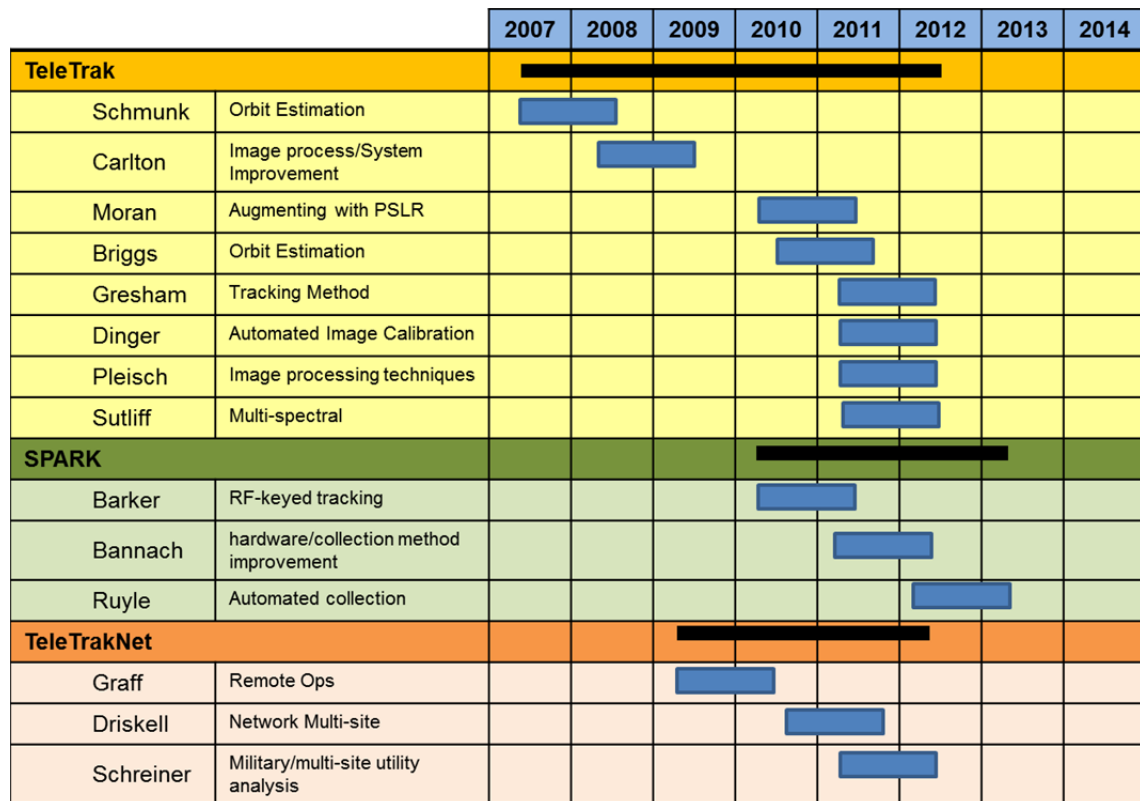


Figure 3: AFIT Tracking Station History Through 2013 [16]

Schmunk successfully used a Meade LX200GPS 10" telescope with a coaxial wide-field digital camera spotting scope to generate initial orbit predictions. Schmunk showed that the Meade mount and integrated GPS receiver was capable of an angular accuracy of 3 arcminutes (at 3σ) and a timing accuracy of ± 0.5 seconds (at 3σ) to create an "angles only" observation [17]. Schmunk was the originator of many of the MATLAB scripts and functions used in TeleTrak. Many of the students following his work on TeleTrak built upon and adapted his code for their applications.

Briggs attempted to improve on the precision of Schmunk's tracking tools and acquire satellites without a priori knowledge with some success in appropriately attributing an acquired satellite to an entry published in a TLE catalog [18]. Briggs adapted Schmunk's code with the "Watcher Track" Graphical User Interface (GUI)

which enabled a “staring” mode that remains pointing at a desired point in the sky and detects satellite streaks. The reported angular pointing accuracy ranged from 2 to 30 arcminutes. This was sufficient for demonstrations in IOD of LEO satellites but was concluded to be well below the precision required for GEO tracking by Moomey, who chose more capable updated astrometry algorithms for his work, thereby relieving the precise angle requirements which cannot be met by most COTS mounts, including the Meade mount. While Brigg’s made significant progress on the TeleTrak groundwork, his code was developed uniquely for his research and was not a true TeleTrak “Version 2.0”.

Gresham attempted to design a real-time optical feedback proportional-derivative (PD) controller using a wide FOV spotting scope [8]. His work significantly improved upon the performance of TeleTrak from 13.1% to 93% of the time when a satellite target is located within 0.05° FOV of the main optic. His research identified limitations in the coupled computer hardware and the internal software of the Meade telescope. While the telescope’s slew rates were sufficient to achieve the objectives, the computational limits of the computer prevented analyzing the video stream at a rate faster than 15 frames per second while communicating with the telescope [8]. Additionally, the telescope’s internal communication system updates its present location accurately to the external computer algorithm approximately every 0.7 seconds [8]. This combined with an update rate of slew speed at a rate of 10 Hz prevented the implementation of a pure proportional controller. Again, the software developed by Gresham was developed as a separate branch of the TeleTrak code and not integrated into a comprehensive software package.

Finally, in 2015, Moomey used an Orion 80mm short tube with a 0.5 focal reducer/field flattener mated to an Astrovid Stellacam II camera on a Meade LX200GPS mount to study the value of GEO SSA with small aperture commercial telescopes [19]. He utilized the existing TeleTrak network in an attempt to compare the performance of a commercial scope to the current Air Force operated optical Space Surveillance Network. He found that TeleTrak was capable of observing low reflectance GEO RSOs as small as 1.5m^2 at a 45° beta angle and 4.75m^2 objects out to 100° beta angle with accuracies of 11 - 17 arcseconds in right ascension and 1.2 - 2 arcseconds in declination [19]. Moomey concluded that these accuracies were on the same order of magnitude as the Air Force's SSN. He was also able to generate GEO TLEs during an observation using TeleTrak in a time span of 5-10% of the time span utilized by the Joint Space Operations Center (JSpOC) to generate its GEO TLEs. Moomey was successful in demonstrating high performance potential for TeleTrak in a specific application of GEO observations.

Following six years of development, the TeleTrak code was centrally stored in a single hard drive organized in folders by each student author or in folders of miscellaneous functions and tools used during TeleTrak research. Many central functions were developed over multiple theses by multiple students. While some documentation exists to explain the process and uses of a function, little documentation exists to piece together the version or final application of a given function. This lack of organization renders the valuable software tools of TeleTrak unavailable to potential users. The varied applications and potential for the TeleTrak tools discussed in this

section further motivate the work to reengineer and reorganize the TeleTrak software to facilitate future research and collaborations both in and outside of AFIT.

Software Design

Reusability is a hot topic in the space industry. Maximizing system lifetime through reusable components is a necessary engineering practice for designing cost effective technologies. The idea of disposing of an airplane after its first flight is absurd, yet it became generally accepted that space vehicles are single use systems (save for a few notable attempts). A lack of reusability is the dominating factor for the cost prohibitive nature of space access which is only now being challenged by industry. In a lot of ways the same “one-and-done” approach is practiced in software design. One aspect of the aversion to reusing previous work is likely cultural, where value is focused on new ideas, and technologies sell based on their novelty over their similarities. Diminishing budgets and a push for responsive development have shifted the focus from performance to cost and efficiency. The contemporary *ad hoc* software design approach restricts modular software reuse and flexible system application. The TeleTrak software provides a prime opportunity to practice reusability through a modular architecture.

It is necessary for the background of this problem to outline the standard approach to software development practiced by software engineers. It should first be noted that many frameworks exist and that no single framework is suitable for every application. A generic development process which appears frequently in the literature will be discussed here to provide a procedural concept. The general approach as

outlined in the open source series: The Linux Development Platform is summarized below in Figure 4 [20].

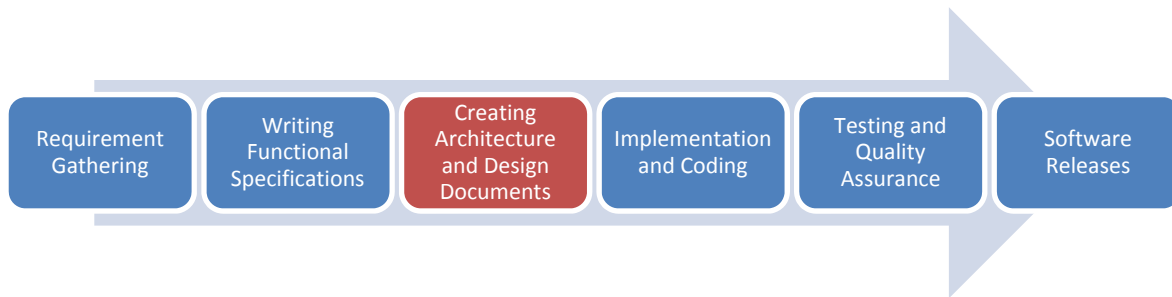


Figure 4: Software Development Process [20]

This process illustrates the basic steps to developing most software tools and is often iterative with customer feedback, debugging, and multiple releases. Highlighted in red is the critical step that is often overlooked by programmers and engineers without a background in software engineering. In the context of TeleTrak, the requirements of the software are to operate a COTS telescope to track an Earth orbiting object. This is broken down into functional specifications. For example, the software must propagate an object's orbit from TLEs, inform the user of visible passes based on a calculated visual magnitude, and command a telescope to track the propagated position of a user specified object. Implementation, coding, and testing took place over several years at AFIT which met these objectives through the development of satellite tracking software tools. As is often the case, creating the architecture and design documents for the software tools was never completed. The implication being that no architecture design or organizational philosophy was established or followed informing future TeleTrak developers how to build upon previous work. This resulted in multiple branching paths and software versions which are incompatible and only meet the evolving research

objectives of each individual developer. Skipping the software architecture and design documentation restricts reusability and future code development.

Similar to the development process, there are countless strategies to the design of the software architecture. The Hierarchical Component Design shown in Figure 5 is one such strategy demonstrated successfully by the SciBox software suite which is borrowed and discussed here.

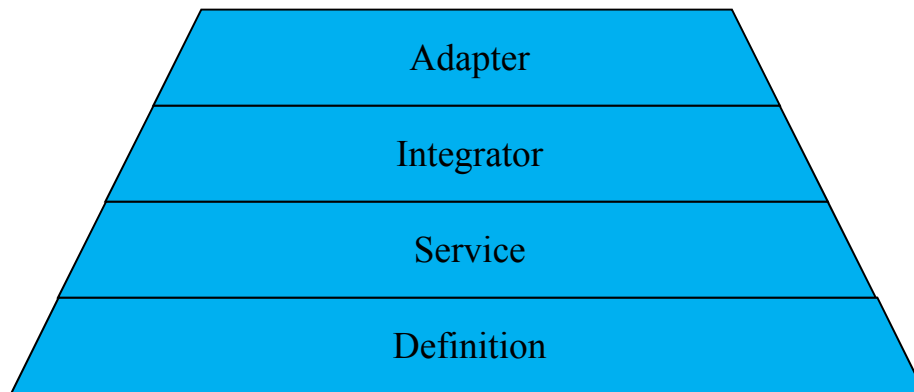


Figure 5: SciBox Hierarchical Component Design [15]

The HCD approach facilitates software reuse and provides an overall organization that is simple and rapidly customizable. The term “component” is often used without explanation. In the context of this research, components are the building blocks that make up the software and exist at all levels of the structure. The architecture of the software structure is defined by the relationships between the components. The principle of the HCD is that all software components are classified and organized by their degree of generality [15]. The most general components are placed in the bottom layer while the most specific components are placed in higher layers. Most importantly, only components in a higher layer can reference a layer below, never in reverse. This prevents circular dependency which is the primary source of non-reusability and a

common occurrence in software design without proper architecture and design documentation. The bottom layer of this structure is the Definition Layer which defines and standardizes the data structures and vocabulary for the program. For example, tracking data for TeleTrak is generated from TLE data reported by NORAD. The TLE format, shown Figure 6, contains all of the necessary orbital elements of an Earth-orbiting object to locate and predict the object's location. Should this format change, the definition layer and all subsequent layers above it in the software hierarchy would be affected.

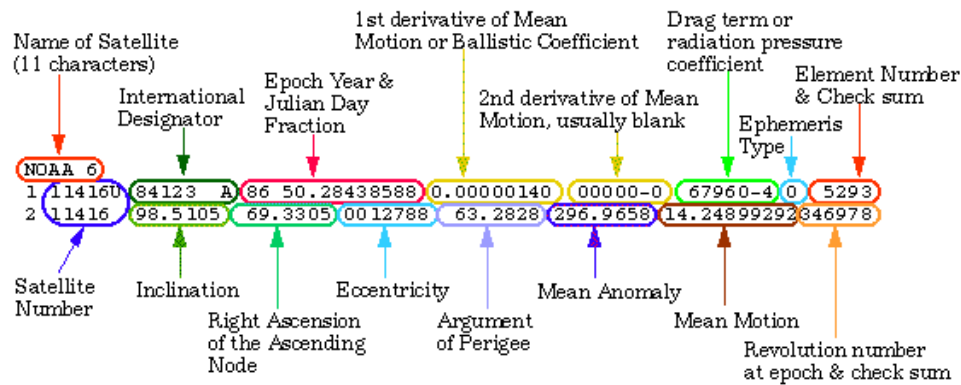


Figure 6: Two-Line Element Set Definition [21]

The Service Layer includes all of the services of the software which operate on the defined data structures. Services include algorithms and input-output functions that the software calls throughout any tasks the software is designed to perform. The Integrator Layer integrates components from the service layer and combines them into higher level services. For example, orbit propagation is a service integrated from reading TLEs, running a propagation algorithm, and converting the results to tracking data. The top layer is the Adapter Layer and is responsible for adapting the collection of services into a configuration environment which the user can access and operate in.

This layer may include a graphical user interface and defines the interaction between the user and the software. It is important to note that each layer may include numerous components, and these components may only reference components across or below its layer, never above. Finally, below all of these layers exists the Virtual Machine Language (VML) Layer which defines the software language and operating environment that the architecture is based in. In the SciBox example, this layer is the Java VML; for the TeleTrak application, the software is based in MATLAB. While this constrains the software architecture, MATLAB is selected because it is the language of previous TeleTrak software, it is a commonly practiced coding language in the aerospace community, and it is well supported on all relevant operating systems and computers for this application.

The HCD architecture can be likened to a carpentry toolbox. The VML Layer is like the box itself which confines the scope of tools that can be developed and stored inside. The Definition Layer is an instruction manual in the box which sets the standards and definitions the tools in the box follow, such as screw sizes, measurement standards, etc. The actual tools in the box represent the Service Layer, such as a hammer or drill which are designed to perform a single, or sometimes more than one, task. The Integrator Layer appears when combining tools to perform a new task. For example, hanging a picture requires the integration of a stud finder, hammer, and nail. Finally the Adapter Layer is represented by the drawers and compartments of the toolbox which defines how the user accesses and interacts with the tools. For one application, a set of tools may be inaccessible or simply ignored by the user, but accessible during a different application.

Modularity is an emergent quality of the HCD due to the organization of services and integrated services which, like a carpenter's tool set, can be modified and substituted as desired. Reusability emerges when substituting a layer or components of a layer from one application for a new application. Since no component ever references a layer above it, the reusable architecture is every layer of the structure below the substituted layer. Designing a toolbox which can only hold a particular hammer would be a foolish endeavor, and as such this is the purpose for only referencing down in the hierarchy. The HCD is a powerful framework for software architecture design which facilitates modularity, reusability, and rapid customization for new applications.

Modularity

According to Microsoft, the term “modularity” refers to the division into a set of functional units (modules) that can be composed into a larger application [22]. A composite application exhibits modularity by allowing the user to access a variety of functions although each function is a discrete module. Modules may interact with one another, but the user only sees an integrated Adapter interface, or “shell”, that appears to operate as a single application. A visual representation of a modularized composite application is shown in Figure 7.

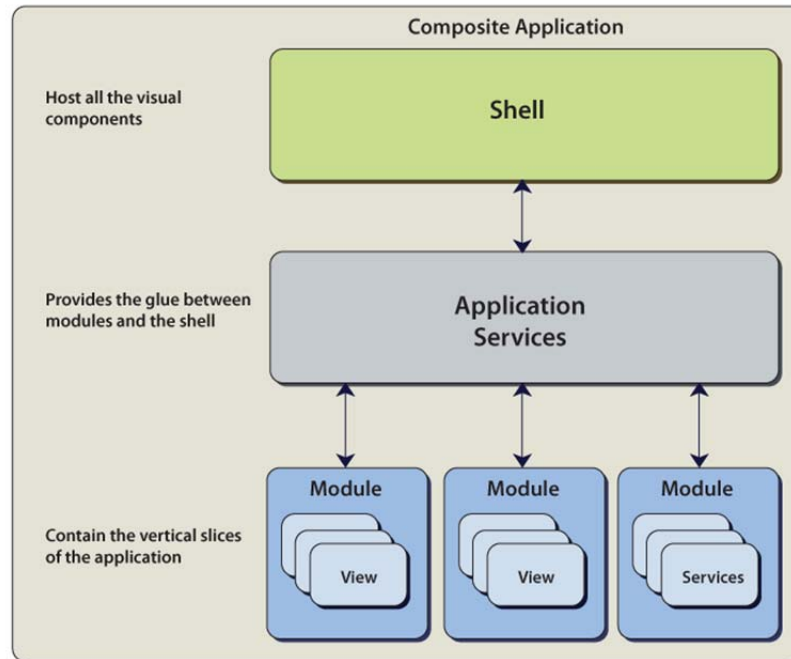


Figure 7: Modular Software Design Concept [22]

Microsoft provides the following guidelines to developing a modular system [22]:

- Modules should be opaque to the rest of the system and initialized through a well-known interface.
- Modules should not directly reference one another or the application that loaded them.
- Modules should use services to communicate with the application or with other modules.
- Modules should not be responsible for managing their dependencies. These dependencies should be provided externally, for example, through dependency injection.
- Modules should not rely on static methods that can inhibit testability.
- Modules should support being added and removed from the system in a pluggable fashion.

Microsoft defines a “Module” as a logical unit in the composite application library [22]. Again, many definitions exist of a software module, but all are intentionally vague. For the purposes of this research, a module is defined as a service or combination of services in the composite application that can be added and removed

from the system and, when combined or called individually, performs the desired tasks of the system.

Although the Microsoft guidelines for developing a modularized system are based in the C# coding language, many definitions exist which reflect the same principles. Simply put, the guidelines for realizing the benefits of modularity in any coding language are to break down a composite application into its distinct tasks, limit dependencies between different tasks, and provide the user with an interface that performs these tasks without knowledge of the behind-the-scenes structure. For example, telescope commanding is a task that the TeleTrak software should perform. The user should have an intuitive interface that enables them to command the telescope without knowledge of the modules or services involved. When a generic command, “Turn Telescope West” is executed, the user should not enter the Meade Telescope string command, “:Mw#”. Rather, execution should be relegated to a left arrow key or other generic button command. In the case of tracking an object, the user should select the target and hit a “Track” button while the software determines which commands to send the telescope to execute the tracking task. In this case, a translator should be implemented which translates system defined commands to the identified hardware language. A translator is a Service Layer component which operates above the Definition Layer translating commands between internal and external definition sets. When new hardware is introduced to the system, the interface does not change, but the modular translating service should be modified or added to include the new hardware language.

Telescope Hardware

For those like this author without a background in telescopes, Figure 8 shows a component diagram of a compound telescope like the Meade LX200-ACF telescope. Figure 9 shows a refractor telescope component diagram like the Orion Short Tube Spotting Scope used during this research. The advantage of the compound telescope is that it uses a combination of lenses and mirrors to focus light in a compact design. This means more power at a smaller size, though typically at a higher cost. The refractor telescope is a traditional tube configuration which is longer and skinnier. They typically provide sharp, high-contrast views at a lower cost, but may need to be significantly larger for comparable power. A smaller short tube refractor telescope is ideal for a spotting scope because it provides a larger FOV to identify and track an object while providing optical feedback to drive the target into the narrow FOV of the main optic.

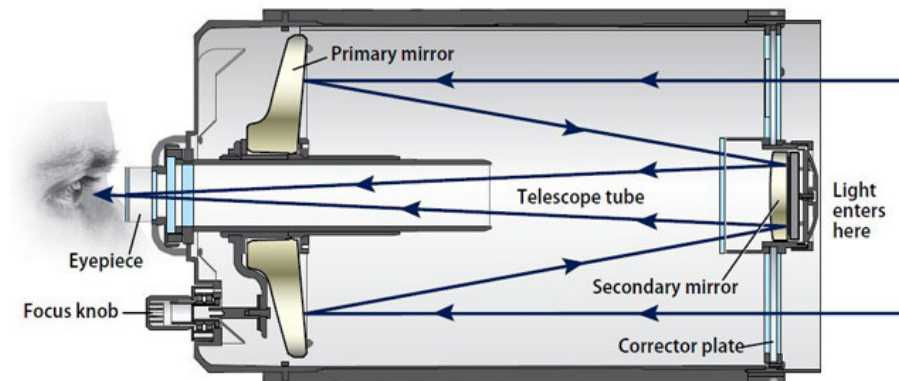


Figure 8: Compound Telescope Diagram [23]

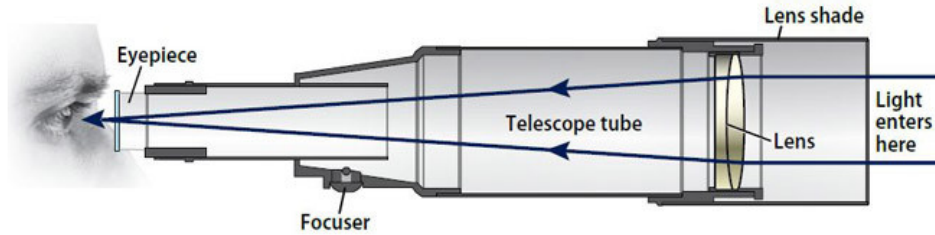


Figure 9: Refractor Telescope Diagram [23]

The mount is the second critical consideration in choosing a telescope configuration. There are three basic types of ground based telescope mounts, the Alt-Az, Dobsonian, and Equatorial mount, shown in Figure 10. The Alt-Az and Dobsonian mounts are the simplest and only pivot up-down (altitude) and left-right (azimuth). The Dobsonian mount sits on the ground and works similar to an Alt-Az mount but is typically paired with a reflector telescope. The Equatorial mount has a single rotation axis parallel to the Earth's axis of rotation enabling it to easily track stars or satellites as they move across the sky. Equatorial mounts are typically used for both refractor telescopes and compound telescopes. Technical details of the telescope equipment used in this research are discussed in Chapter III.

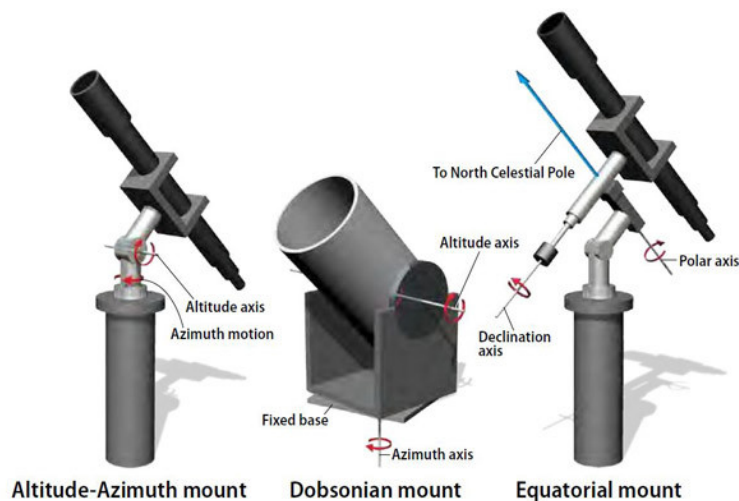


Figure 10: Types of Telescope Mounts [23]

Summary

This chapter identifies significant achievements at AFIT in demonstrating the value of COTS telescopes for SSA; however, like any tool without regular maintenance, software loses its edge overtime. Regaining previous TeleTrak performance and ensuring it for the future is critical to progress SSA research at AFIT and keep the tools sharp and available. The communities of amateur astronomy and academia contains a great deal of knowledge in tracking and observations with commercial telescopes, but the applications to SSA are more unique to the USAF. Recognizing the resources of the community and COTS equipment is important for the DOD to remain cost effective while keeping its edge.

The body of work and research discussed in this chapter is testament to the interest and importance of the topic of augmenting SSA with COTS equipment. The many advantages of low cost, rapidly deployable telescopes for SSA have long been recognized and this research aims to build upon this by reengineering the TeleTrak tools and resources at AFIT.

III. Methodology

Chapter Overview

The purpose of this chapter is to cover the methodology used to reengineer, reorganize, and test the TeleTrak software architecture. First discussed is a description of the TeleTrak equipment and hardware used in this research. Following this will be a section describing the implementation of the Hierarchical Component Design framework and how it affects future TeleTrak users and developers. Building modularity into an existing set of software tools is not a simple task. Modularity emerges from defining and building software architecture as defined in this chapter. Finally, a test scenario is presented to verify and validate the updated software tools through a tracking demonstration.

Equipment

Existing equipment was primarily used for this work. Changes include a new desktop and laptop with Windows 7 operating systems, and an update to MATLAB 2015a. Equipment that was inherited from previous TeleTrak research includes the optics, focal planes, and telescope mounts. The telescope is an Orion short tube with a 0.5 focal reducer/field flattener with an Astrovid Stellacam III camera, attached to a Meade LX200GPS-ACF Telescope on an Alt-Az mount, shown in Figure 11.



Figure 11: Lab Telescope Configuration

The lab configuration, operated from a desktop computer, emulates the same telescope and configuration on the AFIT rooftop operated from a laptop. This enables rapid testing and simulation in the lab environment before testing the hardware and software through observations on the rooftop.

The Orion short tube 80mm refractor telescope shown in Figure 12 is the optic used for tracking and observations in this research. The Orion scope has a native 400mm focal length and $f/5.0$ focal ratio but includes a 0.5 focal reducer yielding a 200mm focal length and $f/2.5$ focal ratio. This scope is designed for wide field-of-view observations and ideal as a guide or spotting scope for the full Meade assembly.



Figure 12: Orion ShortTube 80mm Refractor Telescope Optical Assembly [24]

The Astrovide Stellacam III camera, shown in Figure 13, is installed on the Orion spotting scope. This camera is used because of the low noise Sony HAD CCD sensor and variable gain which allow sharper images with minimal star streaking to capture valid Astrometry solutions.



Figure 13: Astrovide StellaCam III [25]

The CCD has an effective pixel array of 768 (H) x 494 (V) and Unit cell size of $8.4\mu\text{m} \times 9.8\mu\text{m}$ [26], yielding a chip size of 6.45 mm x 4.84mm. Using an effective focal length of 200mm, the field of view of the Orion Spotting scope is calculated using Equation 1 below as 1.85° (H) x 1.39° (V) .

$$FOV (rad) = \frac{Chip\ Size}{Focal\ Length} \quad (1)$$

The 0.5 focal reduction factor on the Orion spotting scope changes depending on the installed point of focus; therefore, the effective focal length is not exactly 200mm. A more accurate determination of the telescope field of view can be acquired through astrometry solutions by uploading a star field image to Astrometry.net, an online open source tool trusted by the astronomy community [27]. Astrometry.net will analyze an image of the night sky containing known celestial bodies and return astrometric calibration meta-data including a list of known objects within the image. A

small variation in the positioning of the focal reducer on the telescope leads to significant error in the calculated field of view of the camera. Furthermore, the video is converted from the native pixel array to an 800 x 600 window; therefore, analyzing an output image from the camera of a star field is likely a more accurate estimate. The results for each telescope and camera reported by Astrometry.net are shown in Table 1. The 10'' Meade uses an F/6.3 reducer. The 16'' Meade telescope is not used in observations for the duration of this research, but is a natural follow on for future work with the software.

Table 1: Telescope - Camera Configuration Properties

Telescope	Orion 80mm	Meade 10''	Meade 16''
Camera	Astrovid Stellacam III	QSI660	QSI660
Field of View (H x V)	1.62° x 1.22°	0.45° x 0.36°	0.18° x 0.144°
Pixel Scale	7.31 arcsec/pixel	1.17 arcsec/pixel	0.468 arcsec/pixel

Finally, the Epiphan VGA2USB Frame Grabber, shown in Figure 14, is used to capture video from the Astrovid StellaCam III [28]. Epiphan Capture Tool software is available online which streams the video from the camera in real time on the screen. Recordings can be made in Audio Video Interleave (AVI) format or PNG, JPEG, and Bitmap (BMP) images. Streaming the video on screen is important for improving the optical feedback which will be discussed in Chapter V.



Figure 14: Epiphan VGA2USB Frame Grabber [28]

Software Architecture

Establishing the software architecture is the first challenge in reengineering a previous set of software tools without architecture and design documentation. The architecture design is limited by the dependencies and organization of the existing components. Circular dependencies and references between services can be designed out of the code in some cases, but in the case of complex co-dependent services with little documentation, the services must either be taken “as is” or redeveloped entirely. Borrowing software tools from outside sources is common practice in which case the tool must include a documented input-output definition for the programmer to adapt to their needs.

Introduced in Chapter II, the Hierarchical Component Design framework is a layered structure which builds from the bottom up with references always from the top down. The benefit of this structure is a simplified architecture which promotes reusability and modularity. The basic principles of the hierarchical architecture are repeated here:

- Components with greater generality are placed lower;
- Components may only reference below, never above;

- Components must contain no circular dependencies.

Software architecture such as the HCD is intended to be constructed during the software development process. The steps listed below used to build the TeleTrak hierarchy are unique to implementing the HCD architecture around existing software.

1. *Identify the system objectives and requirements*

Over the years, the objectives and potential applications identified for the TeleTrak system have expanded greatly. The fundamental requirements of the system are to predict satellite orbits based on the TLE standard and accurately point a Meade telescope at the target as it passes over the ground site. For the purposes of this research, these fundamental requirements are addressed with a focus on a generalized approach to facilitate expansion.

2. *Identify the system services*

This step identifies the services the software must perform to fulfill the objectives and requirements. For existing software, the services of the system may or may not be distinguished in the existing structure and must be identified individually. An organizational concept identifies hardware constraints where services are mapped to specific hardware and present opportunities for generalized dependencies.

3. *Remove unnecessary dependencies when possible*

Circular dependence occurs when separate services directly or indirectly reference one another. This will not impact the system modularity if circular dependencies occur within discrete modules, but minimizing dependencies among services maximizes the component structure breakdown. The goal is to

generalize how services acquire their data. For example, the output of the camera device (i.e. pixels) should be referenced instead of searching for the device drivers specific to the hardware. Removing circular dependencies is increasingly difficult for more complex systems, but can be accomplished through dependency injection or observer patterns which are beyond the scope of this research. Fortunately, many dependencies are avoided in MATLAB by the division of functions which maintain internal variables. TeleTrak dependencies that constrain the system exist between the hardware, software, and string callbacks within the TeleTrak_v8 GUI. A string callback is a callback function internal to a GUI written programmatically. Without distinguished callbacks, callback functions are dependent on the input-output arguments of the executed script rather than variables native to the independent functions. String callback functions can be modified but cannot be removed or replaced, making them non-modular.

4. *Classify the services*

Once the services that make up the architecture are identified, they need to be organized and classified by degree of generality. Services without dependencies or callbacks to other services are the most general. Services that combine lower level distinct services are less general, and will be placed higher. The distinction between services into separate MATLAB functions or callbacks is critical to modularity and constructing the hierarchy.

5. *Design the user interface*

Perhaps one of the most important steps in the software design process is building the user interface. The interface defines which services are accessed and how the user interacts with those services and is the final application specific layer to the architecture. An intuitive interface that gives feedback to the user on incorrect inputs or errors encountered during operation is critical to the success of the software. The TeleCalc interface is designed to follow these principles with a large input and status log window, simplifying the process of generating the desired tracking data.

6. *Construct the layers of the hierarchy*

Once the individual and integrated services of the software are identified, distinguished, and classified, and one or more interfaces are designed for the user to interact with the software, all of the components can be organized into the component hierarchy. The five essential layers are the Virtual Machine Language, Definition, Service, Integrator, and Adapter Layers. Figure 15 illustrates the HCD of the TeleTrak software. The complete list of services involved in the total TeleTrak architecture are too numerous to display in a single figure; rather, Figure 15 provides the reader with an illustration of how the hierarchical structure is built followed by a description of each layer.

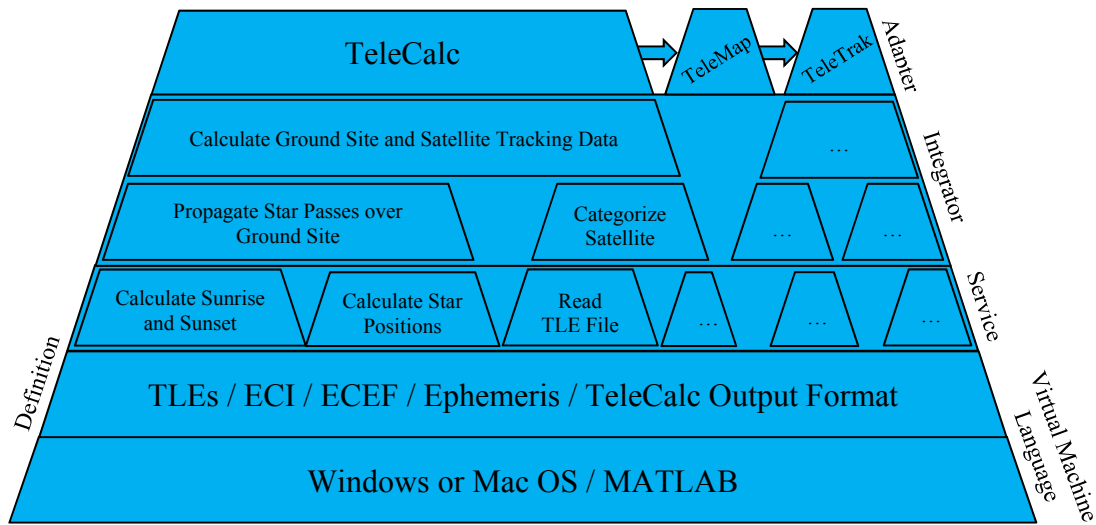


Figure 15: TeleTrak Hierarchical Component Design

Virtual Machine Language Layer

The bottom layer of the TeleTrak architecture, the Virtual Machine Language, is defined as the MATLAB coding language operating in either the Windows or Mac operating system. Specifically, the software is updated in MATLAB 2015a to operate in both a Windows 7 and Mac OS X environment. The TeleCalc and TeleTrak_v8 GUIs necessary to produce tracking data and conduct tracks are both tested and operate when run from MATLAB 2012a, but have not been tested in earlier versions. The significance of Virtual Machine Language layer is that changes that occur in this layer will likely be fatal to the entire system or result in unforeseen or undetectable changes to operating performance. Operating in a well-established coding environment such as MATLAB provides some reduced risk and robustness because it is has interest in maintaining support for all common future operating systems and for previous versions of its own software. Nevertheless, compiling the TeleTrak code into executable files circumvents this issue and will be discussed later.

Definition Layer

The second layer to the TeleTrak architecture is the Definition Layer which sets and maintains the standards for the entire system such as input-output data structures and standardized definitions in the space community. The most significant definitions which the TeleTrak hierarchy borrows from are the TLE format, time standards, ephemeris data, and reference frames. Like all space operations, standardizing time is critical to satellite tracking. The Julian Date standard is used through all orbit calculations and Zulu time is set as the reference time for TeleTrak observations. In this case, ephemeris data refers to the position and velocity of an Earth-orbiting object with respect to its reference frame.

Three reference frames are used for basic TeleTrak tracking, Earth Centered Inertial (ECI), Earth Centered Earth Fixed (ECEF), and South-East-Up (SEZ). The ECI frame is defined by its origin at the center of the Earth's center of mass, the x-axis pointing to the vernal equinox, and the orthogonal y-axis falling on the plane of the equator. The ECI frame is used for ground site coordinates and satellite position propagations. The ECEF frame is also defined with the origin at the center of the Earth, but with the x-axis pointing towards the Greenwich Meridian, and the y-axis completing the equatorial plane. The ECEF frame rotates with the Earth and is useful for plotting satellite positions from a ground station-centric point of view. The SEZ frame is a topocentric coordinate system with an origin at a ground site on the Earth's surface, the primary axis pointing south, secondary axis pointing east, and the third axis pointing zenith, perpendicular to the horizon. Converting to azimuth and elevation of a target in the sky is advantageous in the SEZ frame, as shown in Figure 19, and thus the SEZ

frame is necessary for simplifying telescope pointing. Small variations of the SEZ coordinate system such as the Up-Right-Downrange (URD) coordinate system have been used in previous TeleTrak research to define a sensor frame of reference and convert video measurements back to the SEZ frame for the purposes of orbit determination.

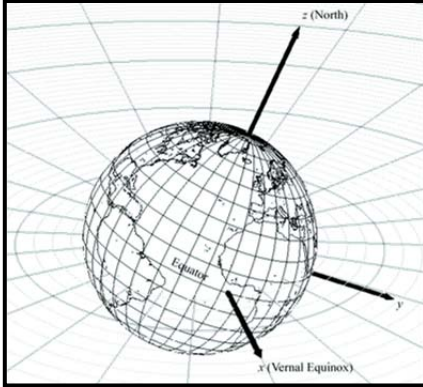


Figure 16: Earth Centered Inertial Frame [28]

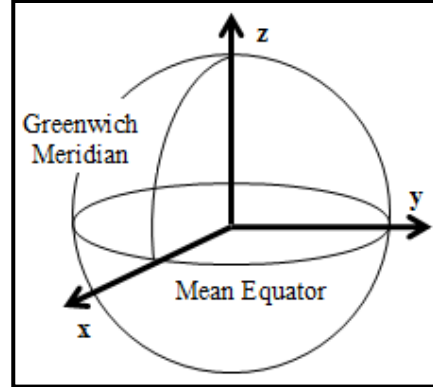


Figure 17: Earth Centered Earth Fixed Frame

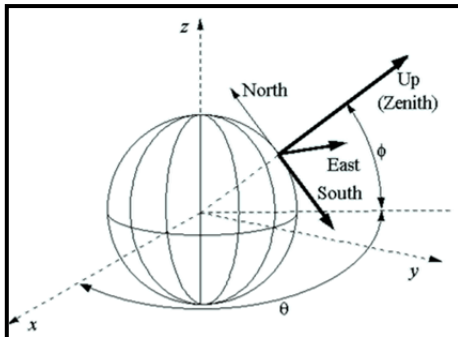


Figure 18: South-East-Zenith Frame [28]

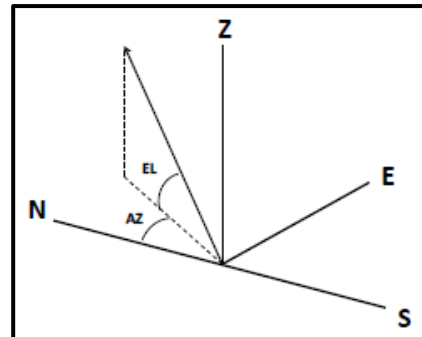


Figure 19: SEZ Reference Frame Relation to Az-El [17]

Another set of definitions included in the Definition Layer are the referenced data files in TeleCalc. The services that reference each data file and the data file descriptions are given in Table 2.

Table 2: Reference Data File Descriptions

Reference File	Service	Description
TLE File	watcher_precalcs.m	NORAD Two-Line Element Set Input File
catalog.dat	killer_getstars.m	Yale Bright Star Catalog (BSC) file containing bright reference stars
circ179_nutation_terms.mat	killer_getstars.m	Earth nutation terms to increase star position accuracy
sun.txt	getsun.m	Sun ephemeris data from 2015-Aug-20 to 2017-Aug-20 (requires updating)
ha_gps.txt	getcatalogsats.m	Heavens above brightness file of known satellite visual magnitude data
mcnames.txt	getcatalogsats.m	Molzcan brightness file of known satellite visual magnitude data
qs.mag	getcatalogsats.m	McCants brightness file of known satellite visual magnitude data

The final definition set by TeleTrak is the output tracking data structure. The output file of TeleCalc.m is a “*.mat” file formerly known as “precalc_results.mat”. The output data file name and path are now defined by the user. This data structure is referenced when mapping orbits with the TeleMap.m tool and when operating the telescope from the TeleTrak_v8.m GUI. Changes to the data structure will cause both of these programs to fail, so maintaining the same definition is critical. The user does not need to know the details of the data structure, but a developer looking to make improvements should understand how the data is stored. For this purpose, the output variables and a description of each are given in Table 3.

Table 3: TeleCalc Output File Definitions

Variable	Description
databaseindex	Index of object ID values which meet user criteria
height_alt_gps	GPS altitude of user selected ground site location
height_geoid_gps	Geoid height of user selected ground site location
lat_local_gps	Latitude of user selected ground site
long_local_gps	Longitude of user selected ground site
passdatabase	Includes all object position and time data in SEZ frame for each object passes that meet user criteria between sunset and sunrise of the user selected observation night
r_ECEF_vec	Contains x,y,z position data (km) in ECEF frame and Julian Date time stamps for each object from sunset to sunrise of the user selected observation night (used for mapping)
r_ECI_vec	Contains x,y,z position data (km) in ECI frame and Julian Date time stamps for each object from sunset to sunrise of the user selected observation night (used for mapping)
Sat_lla	Contains object name, ID, Latitude (deg), Longitude (deg), and Altitude (m) between sunset to sunrise of the user selected observation night (used for mapping).
sat_splintype	Defines spline type used for interpolating target's position and rates (6-cell for satellites/arcs or 7-cell for stars/rasters)
satindex	Contains object ID, Name, Standard Magnitude, and source of Standard Magnitude data for objects that meet user criteria
sscindex	Contains object ID of all objects in TLE file
star_slinetype	Defines spline type used for interpolating target's position and rates (6-cell for satellites/arcs or 7-cell for stars/rasters)
stardatabase	Includes star position and time data in SEZ frame for bright stars that appear in the night sky between sunset and sunrise of user selected observation night
stardatabaseindex	Star ID numbers of all program selected bright stars
stars	Star ID number and brightness data for all stars in Yale Bright Star Catalog (BSC) considered for "stardatabase"
vmag	User Selected minimum visual magnitude threshold criteria used for down selecting from TLE file

The “passdatabase” variable is a cell structure containing all of the tracking data and deserves greater explanation. The first column of the structure is the database index. Each row of the second column is a cell of data for a full pass over the specified ground station. If a satellite has more than one pass over the ground station that meet the user’s criteria, then an additional cell of pass data will be stored in subsequent columns. The pass data in each cell is, from top to bottom, starting Julian Date of pass, ending Julian Date of pass, Azimuth, Elevation, Range, Azimuth Rate, Elevation Rate, and Range Rate.

Service Layer

Placed above the Virtual Machine and Definition layers is the Service Layer which includes all of the independent services that the software performs. These are standalone functions that operate on the data structures in Table 2 and perform the necessary routine algorithms to produce the tracking data.

Integrator Layer

The Integrator Layer takes the independent services such as reading a TLE file, calculating object orbits, and determining object visual magnitude and combines them to higher level services such as categorizing TLE objects to meet user criteria. The Integrator Layer may have multiple levels where integrated services are combined to even higher level services, such as “watcher_precalcs.m” which integrates all of the TeleCalc services into a single script file which the Adapter Layer TeleCalc GUI operates from.

Adapter Layer

The final layer, the Adapter Layer, contains the user interfaces that enable the user to interact with and execute the system's services. The TeleCalc adapter collects the user data inputs for the desired type of tracking for a night's observations. The purpose of this layer is to provide an intuitive interface, shown in Figure 20, for the user to set the observation conditions and produce the desired tracking data.

TeleCalc Inputs	
Geodetic Latitude (deg)	39.782161
Geodetic Longitude (deg)	-84.083893
GPS Altitude (m)	270
Geoid Height (m)	-33.88
Start Date of Observations (dd-mm-yyyy)	21-Jan-2016
End Date of Observations (dd-mm-yyyy)	22-Jan-2016

Preset Parameters	
Minimum Satellite Period (min)	44
Maximum Satellite Period (min)	225
Minimum Satellite Brightness Threshold	5.0
Elevation Angle Threshold (deg)	10.0

Presets

LEO MEO GEO

AFIT Lab AFIT Roof Big Guns

Time Steps
Orbit Time Step (s): 10
Pass Time Step (s): 0.5

TeleCalc Status Log

Help Run TeleCalcs Restore Defaults Clear Log Close

Figure 20: TeleCalc Interface

The Adapter Layer is also responsible for much of the robustness of the system by communicating to the user the system limitations and enforcing the boundaries to prevent program crashes or generating erroneous data. This is accomplished in the TeleCalc interface with a Help window and value boundary prompts for each input box when holding the mouse over the description. Input values include geographic properties of the ground site, start and stop date of observations (spanning a single night), satellite period constraints, minimum satellite brightness threshold, and ground

site elevation angle threshold. Preset buttons are provided to set inputs to LEO, medium Earth orbit (MEO), or GEO satellites from three AFIT locations. Finally, the user can set the orbit propagation time step between discrete satellite locations, and the time step during a pass over the ground site used for telescope pointing. At default settings for a TLE file of approximately 100 satellites, a full set of calculations takes approximately 30 seconds, so calculation run times are not a constraint prior to a period of observations. Data file sizes are potentially the only constraint and are affected by the user defined orbit time step and the number of satellites in the TLE file that meet user criterion. A time step during a pass of one second has been shown to be “good enough”. The TeleCalc tool is developed by updating and reconfiguring the existing MATLAB software that conducted tracking calculations and building an Adapter Layer above it. The second Adapter is the TeleMap tool which interprets the TeleCalc results into a visual interface for object mapping and mission planning. TeleMap is developed from the ground up for the TeleTrak system and includes three tabs: World Map, Ground Site Map, and 3D Globe, shown in Figure 21, Figure 22, and Figure 23 respectively.

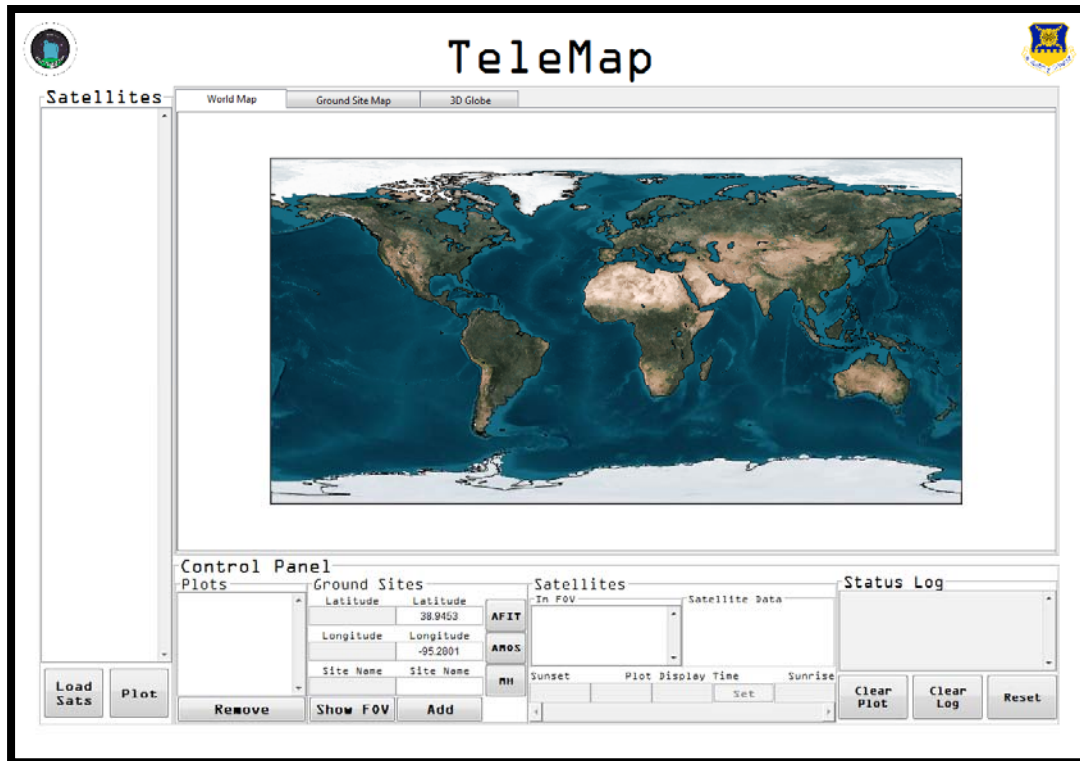


Figure 21: TeleMap Interface

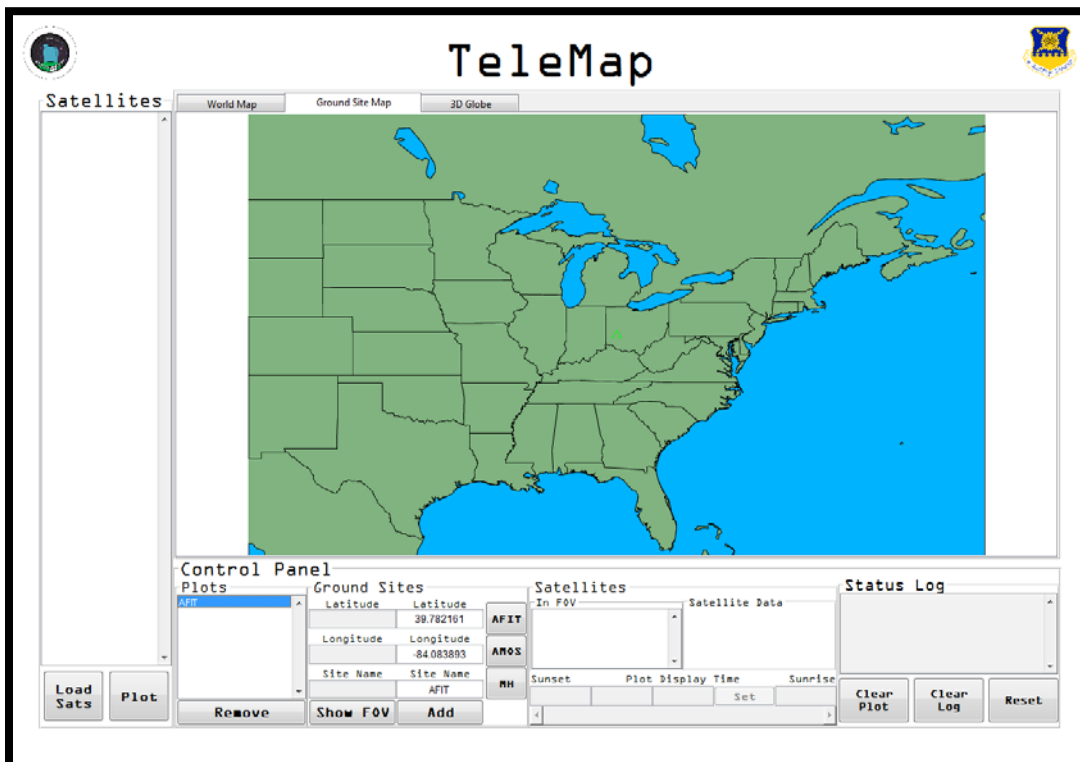


Figure 22: TeleMap Ground Site Tab

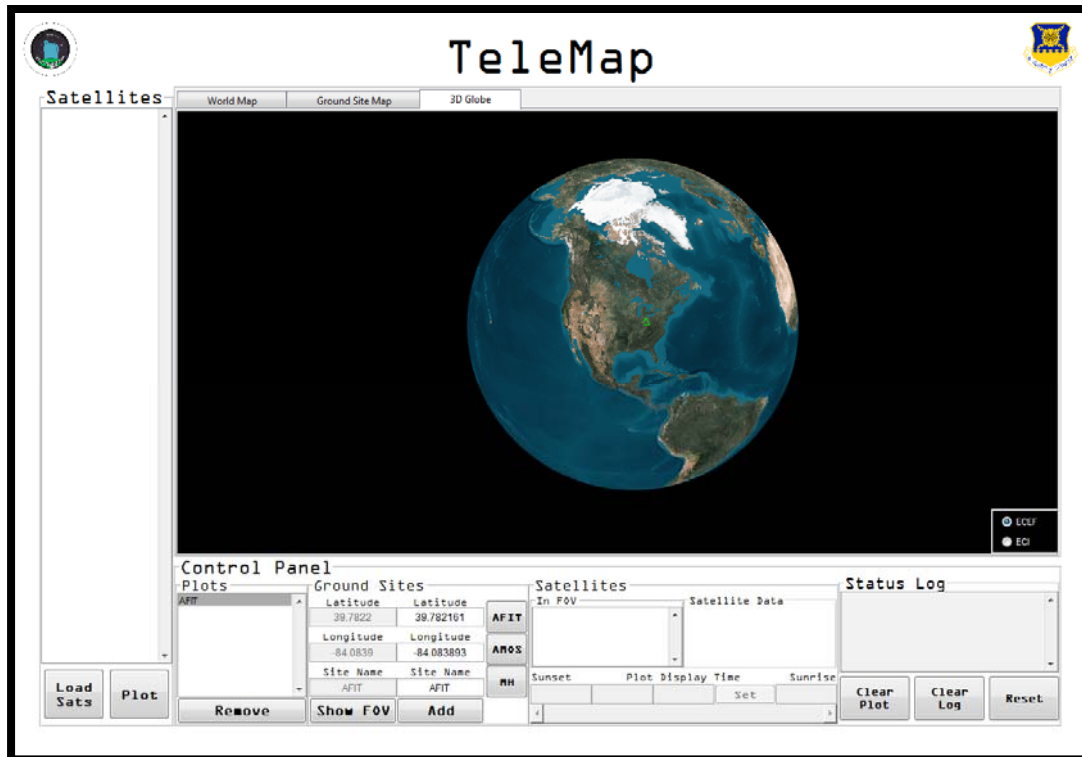


Figure 23: TeleMap Globe Tab

The third and final adapter is TeleTrak_v8. This is the interface used for operating the telescope during observations. The TeleTrak_v8 tool is updated from the existing “watcher_trackgui_v6.m” file by fixing errors preventing it from operating without the previous operating system and camera equipment. The TeleTrak_v8 interface, shown in Figure 24, loads the user selected tracking data from TeleCalc and updates loaded star and target positions on-screen in real time according to the computer’s clock. For this reason, operating from a computer with either internet access or an updated accurate clock is critical.



A dependency diagram is necessary to visually identify the hierarchy of the components in the structure. A full TeleCalc dependency diagram is shown in Figure 25. Each component below the TeleCalc Adapter Layer is a service that, when combined together, performs all calculations necessary for satellite tracking. Highlighted in green are the higher level integrated services which combine the services below in their chain. An arrow from the “TLE File” to the “TeleCalc.m” adapter represents a required input TLE file of either a “.txt” or “.tle” file type. The “*.mat” output file represents the generated tracking data with a user specified name. Appendix D contains a collection of the TeleTrak software architecture documents.

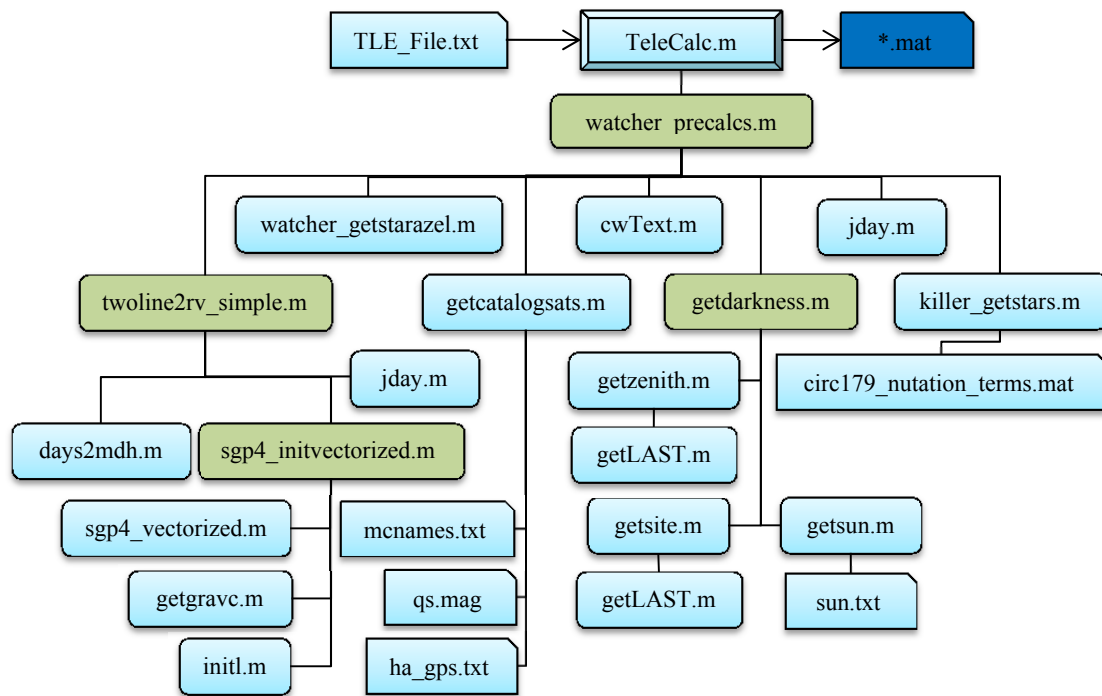


Figure 25: TeleCalc Dependency Diagram

Modularity

The available definitions of modularity and modules discussed in the background section are consistently vague. The definition of modularity established in Chapter II is the degree to which a composite application is composed of modular components. The definition of a module is then established as a service or combination of services in the composite application that can be added and removed from the system and, when combined or called individually, performs the desired tasks of the system. These definitions identify three different layers of modularity which fit in to the Hierarchy Component Design, the Service Layer, Integrator Layer, and Adapter Layer. Figure 26 illustrates an example of the Service, Integrator, and Adapter Layer modules within the TeleTrak architecture.

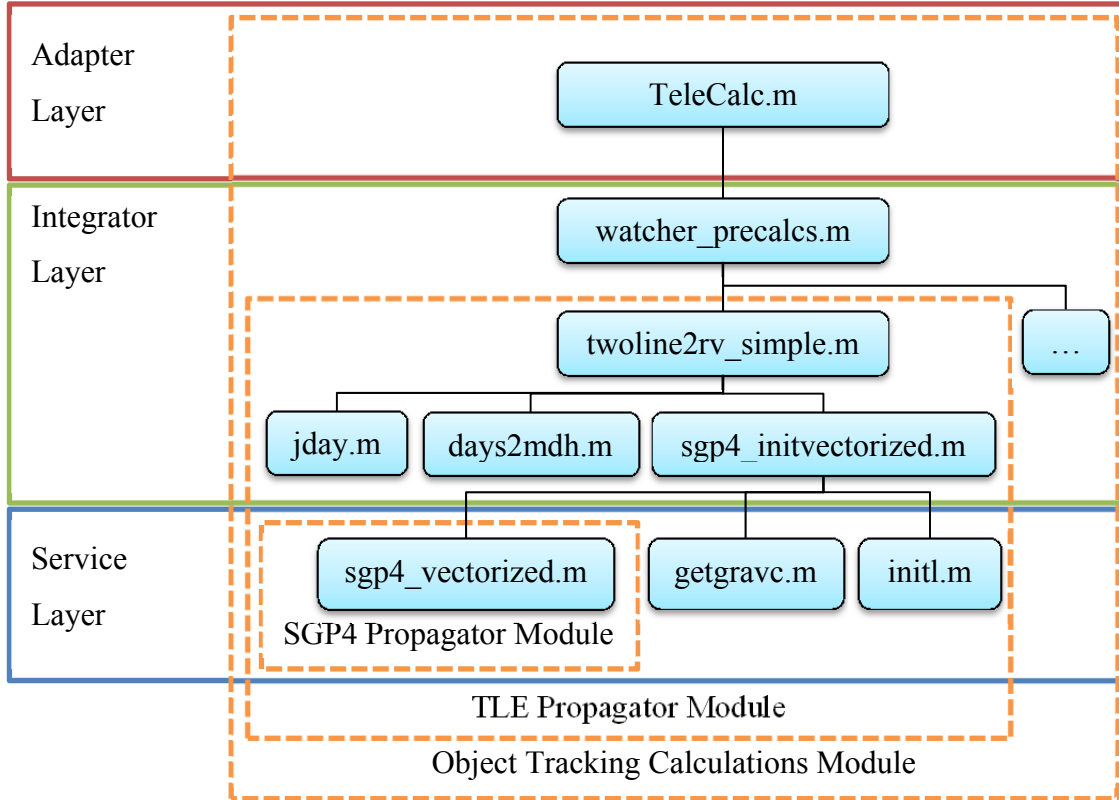


Figure 26: TeleTrak Modularity

In this example, the Service Layer module is responsible for propagating an orbit using the Simplified General Perturbations-4 (SGP4) model. “sgp4_vectorized.m” is a MATLAB version of the SGP4 routine adapted by Beck from code originally written by Vallado, and “vectorized” by Schmunk [17]. If a future user wishes to use a different simplified perturbations model, the SGP4 module can be substituted provided the same input-output definitions established by the Definition Layer of the architecture are maintained. The TLE Propagator Module is an Integrator Layer module which includes the integrated service “twoline2rv_simple.m” and its three referenced services “jday.m”, “days2mdh.m”, and “sgp4_initvectorized.m”. Also included in the TLE Propagator Module are the three referenced services of “sgp4_initvectorized.m”, which

are “sgp4_vectorized.m”, “getgrav.m”, and “initl.m”. While more modules exist at the Integrator Layer, these functions are exclusively part of the TLE Propagator Module. A single module may be called multiple times during a task, but services within a module will not be called by multiple modules. In other words, services are only part of a single module unless they appear in multiple instances, such as a “jday.m”, a Julian Date conversion. Finally, the Object Tracking Calculations Module at the Adapter Layer encapsulates all of the calculations necessary for object tracking and produces a single “*.mat” output file with a file name and path specified by the user. This module is at the Adapter Layer because it is expressed as the “TeleCalcs.m” GUI which adapts all of the behind-the-scenes callbacks and processes in a single user interface. TeleCalc is the first step to tracking objects with TeleTrak and exhibits modularity because it combines the services in the composite application to produce a desired data file and it can be substituted from the system by another module that performs the same task in a different method so long as the input and output of the module is based in the same bottom Definition Layer. Maintaining the same definition is critical because it is the variable names and reference language that the follow on orbit mapping and object tracking applications rely upon.

Testing

One of the primary objectives of this research is regaining previously demonstrated tracking capability with the TeleTrak system. A test scenario for a night of observations is necessary to verify and validate the performance of the software tools at the conclusion of this research. The test scenario is planned using the same telescope

and ground station hardware as previous research, albeit with an update from the StellaCam II to StellaCam III device, in conjunction with the updated software tools for observations on the night of 5 January, 2016. The test uses an open-loop configuration which points the telescope to an orbiting object's propagated position from the TLE file throughout the pass. Test conditions are described below.

Tracking test data is generated from a TLE file of 132 of the brightest tracked objects using TeleCalc. The TLE test file is reported by NORAD on 5 January, 2016 and used for observations on the same night and following morning. The propagated LEO objects are primarily rocket bodies and inactive satellites but also include the International Space Station (ISS) and the first Chinese Space Station, the Tiangong-1. Test data is also analyzed using TeleMap and STK to predict pointing accuracy. The telescope is located on the AFIT rooftop and commanded by a laptop running TeleTrak_v8 in MATLAB. A remote desktop connection is used from the AFIT ground station to the laptop, shown in Figure 27 and Figure 28 respectively. The observed targets and results of the test demonstration are discussed in Chapter IV.



Figure 27: AFIT Rooftop Telescope Configuration

Remote Desktop

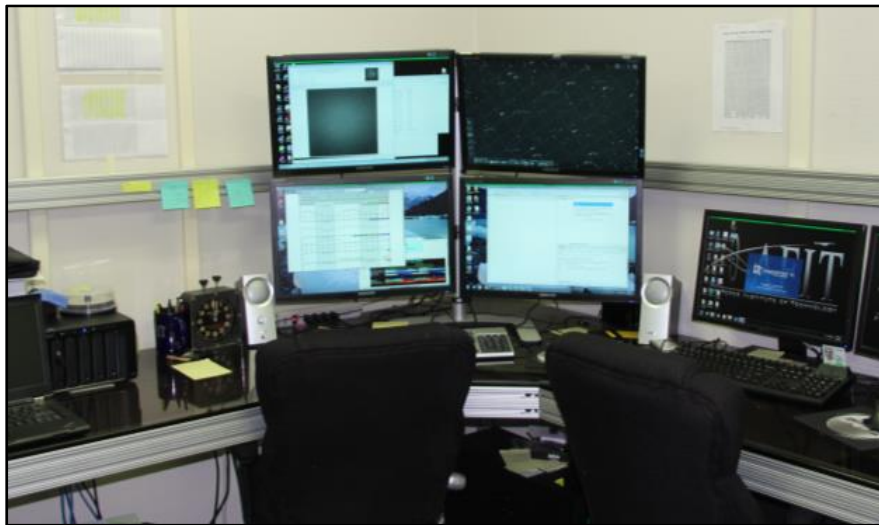


Figure 28: AFIT Ground Station

Summary

Chapter III discusses the equipment, software, and testing that is involved in this research. The equipment, including the telescopes and camera hardware, is primarily inherited from previous research except for an upgrade to a Windows 7 machine with MATLAB 2015a. This chapter also details the process used to implement the Hierarchical Component Design and the emerging modularity. The software user will likely have no interest in the underlying software framework; rather, it is designed to inform TeleTrak software developers how to make modifications, remove and add modules, and build upon the architecture for various applications. Finally, a tracking test scenario is presented to verify and validate the updated software tools.

IV. Results

Chapter Overview

Optimism is a necessary trait when testing software in an environment where success and failure is determined by a second or a single bug in thousands of lines of code. Working almost exclusively with software without getting to see it in action would be devastating, and the Ohio weather in the winter can be unforgiving to say the least. Fortunately the Dayton skies cleared, and the software was successfully put to the test. In this chapter, the TeleCalc test data is first compared against STK results using the same SGP4 orbit propagation model followed by an analysis and discussion of observation test results.

Test Results

The following TLE set for the ISS reported by NORAD on 5 January, 2016 is propagated using the TeleCalc SGP4 routine and compared against the SGP4 model in STK.

ISS (ZARYA)

1 25544U 98067A 16005.60834383 .00006461 00000-0 10054-3 0 9997

2 25544 51.6422 153.8395 0008393 4.0361 101.9119 15.55262591979539

Figure 29 illustrates the difference between the calculated x, y, z position of the ISS in the ECI frame for one full orbit starting from the Orbit Epoch of 14:36:00 UTC on 5 January, 2016 using TeleCalc and STK. The Euclidean Distance, d , represents the straight-line distance between the two points in space calculated by Equation 2.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (2)$$

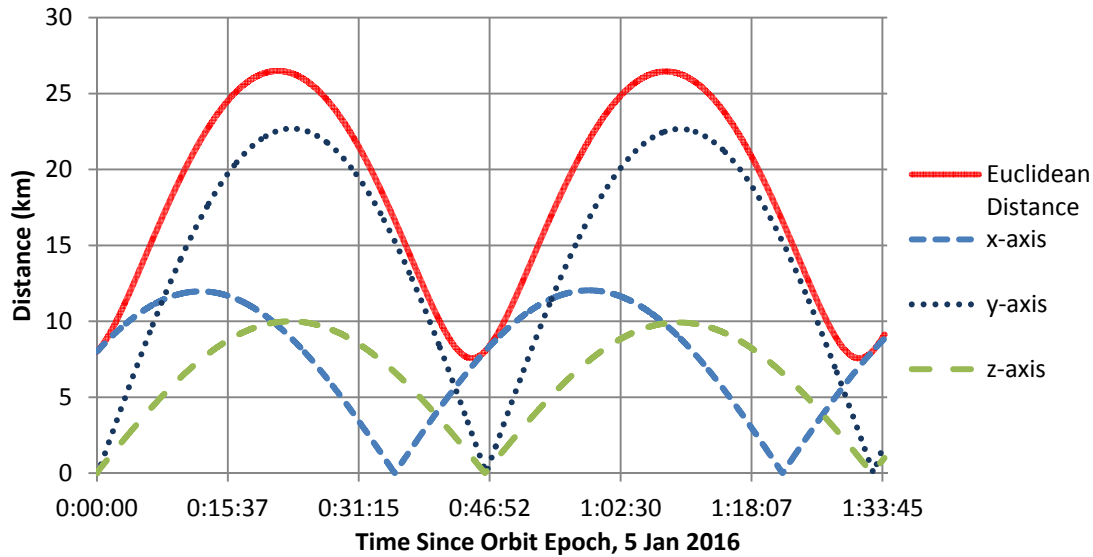


Figure 29: Difference between TeleCalc and STK ISS Position for Single Orbit

Two details are immediately apparent in this comparison. First, the difference between the models is unusually high. Since the results are using the same SGP4 model, the difference in results is expected to be on the order of meters rather than kilometers. Second, the difference in position between the TeleCalc and STK results is periodic. Plotting the two orbits in TeleMap shows that the maximum difference occurs near the equator, and the minimum difference at peak latitude. The difference is plotted beginning at Orbit Epoch because they should be starting from the same position; however, Figure 29 shows a disparity of 8.00 km between the two models for the initial position of the ISS which is almost entirely in the x -axis. The initial point according to each model is plotted in TeleMap in Figure 30, verifying that the disparity is not a result of a difference in time. Next,

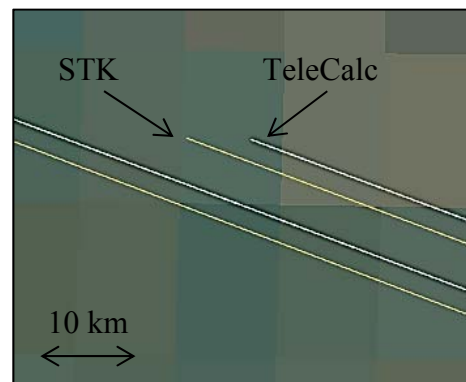


**Figure 30: ISS Orbit in TeleMap
STK vs TeleCalc**

Figure 31 shows a comparison of the magnitude of the ISS position vector according to each model.

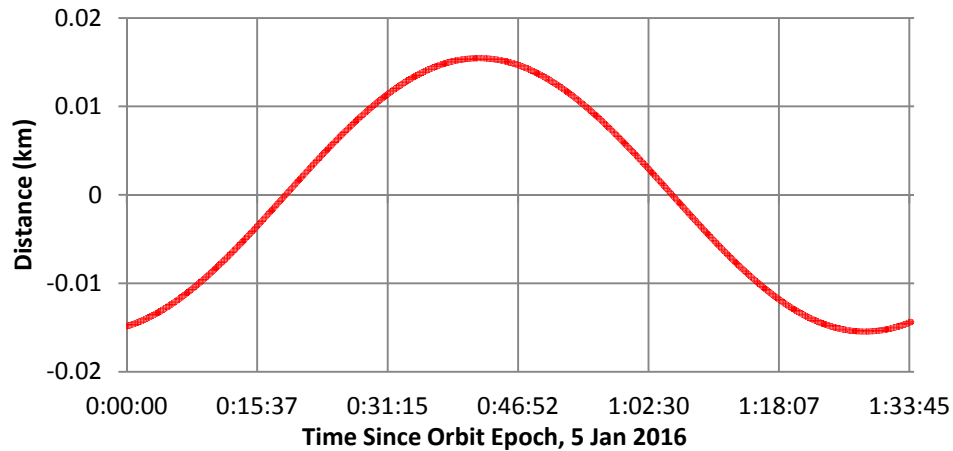


Figure 31: Difference between TeleCalc and STK ISS Position Vector Magnitude

The difference between the ISS position vector magnitudes remains less than 15m which means the orbit altitude with respect to the center of the Earth is consistent between the two models. To verify that the initial displacement exclusivity to the x-axis is coincidental due to the high latitude of Orbit Epoch,

Figure 32 shows the calculated displacement between the two models from Orbit Epoch carried out for a full day.

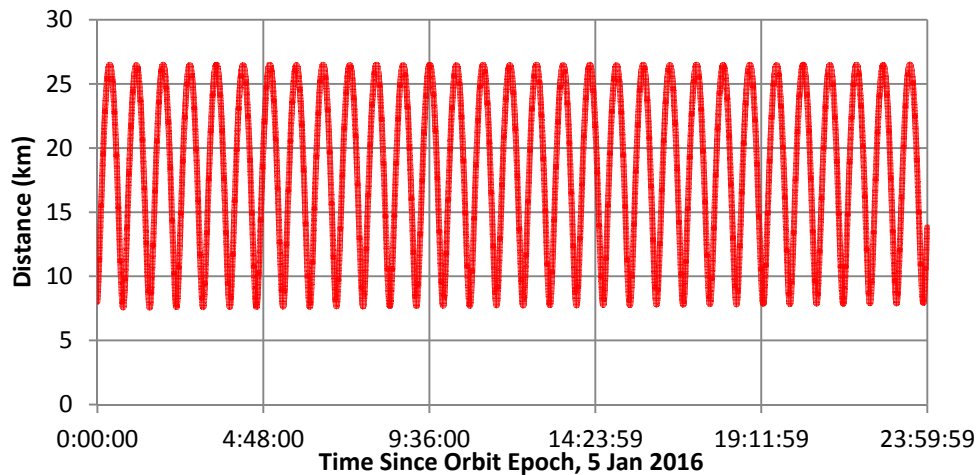


Figure 32: Distance between ISS Position According to STK and TeleCalc

The oscillatory nature of the displacement according to the two models appears to remain consistent with an unchanging average displacement of approximately 17.5 km. These two characteristics are expected of two propagators using the same algorithm from different starting points. One possible source of the error is that the STK and TeleCalc SGP4 models are interpreting the position of the satellite from the TLE data differently, although no evidence has determined this to be the case. The most likely source of the disparity is that the MATLAB and STK propagators are using a different inertial reference frame, although each propagator claims to be based on J2000 using the WGS84 gravity model. It should be noted that neither of these models are “truth” values, but rather comparisons of the MATLAB SGP4 model used in TeleCalc with the STK SGP4 industry model.

Ultimately the significance of the propagation model accuracy for this research is the impact on the pointing requirement of the telescope. As determined during the methodology, the field of view of the spotting scope is $1.62^\circ \times 1.22^\circ$. Assuming the telescope points perfectly to the specified location, and acknowledging inherent errors, the predicted target location will have to be accurate to at least 0.81° in azimuth and 0.6° in elevation to be acquired. The worst case scenario for the example orbit would be for a pass directly over a ground site when the maximum displacement between the models of 26.5 km occurs. For a ground site at sea level directly nadir of the ISS at an altitude of 400 km, the maximum difference in elevation is approximately 3.7° , far beyond the 0.61° threshold for acquiring the target. Comparing the calculated telescope look angles using TeleCalc and STK data in Figure 33, the greatest difference in the models for the ISS pass over AFIT is 0.943° in elevation and 1.276° in azimuth.

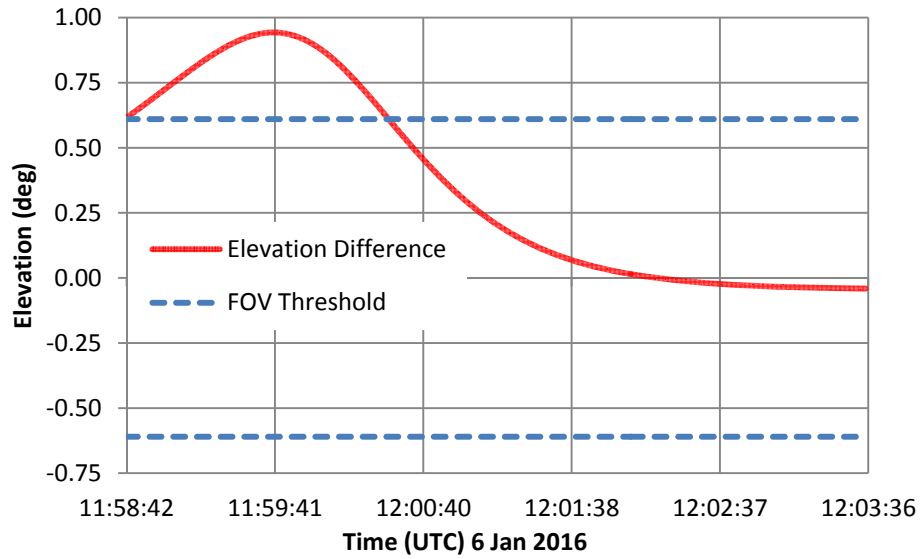


Figure 33: Difference between TeleCalc and STK ISS Elevation Angle from AFIT

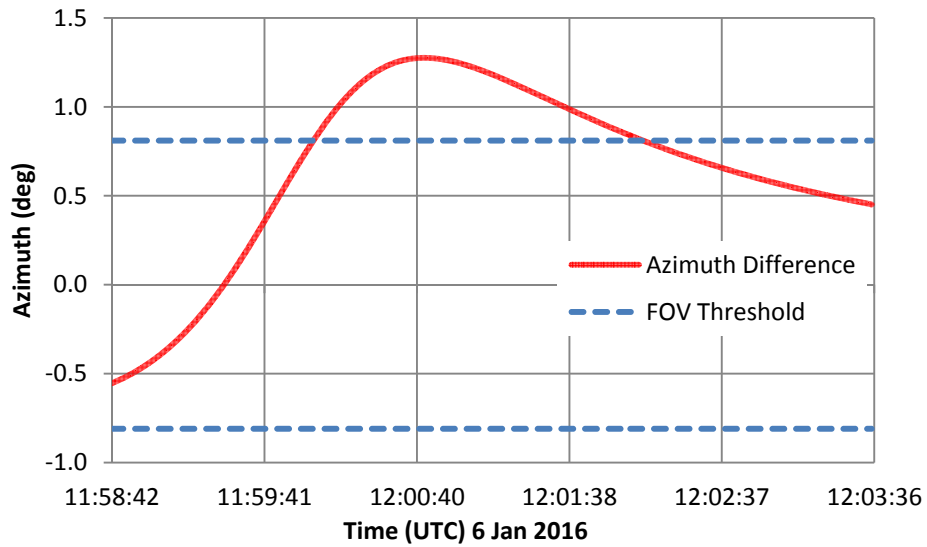


Figure 34: Difference between TeleCalc and STK ISS Azimuth Angle from AFIT

The results shown in Figure 33 and Figure 34 demonstrate a significant impact on acquiring the target based on the model used to calculate the telescope pointing angles. The maximum disparity between the models temporarily exceeds the FOV thresholds of the spotting scope in both azimuth and elevation. The Az-El predictions from the TeleCalc propagation tool are used to observe the targets in this research. If

the STK values are taken as truth, then the ISS should not appear in the FOV of the spotting scope from the start of the pass until 12:02:00 UTC when the STK predicted values are outside the field of view of the spotting scope. As the rest of the results will show, the ISS is captured, but not in the center field of view.

As shown previously, the ISS makes a pass over the AFIT ground site at an elevation angle greater than 10° from 11:58:42 to 12:03:36 UTC on 6 January, 2016. The pass is plotted in TeleMap using the TeleCalc results in Figures 35, 36, 37 and 38.



Figure 35: ISS Pass Start 11:58:42 UTC



Figure 36: ISS Pass End 12:03:36 UTC

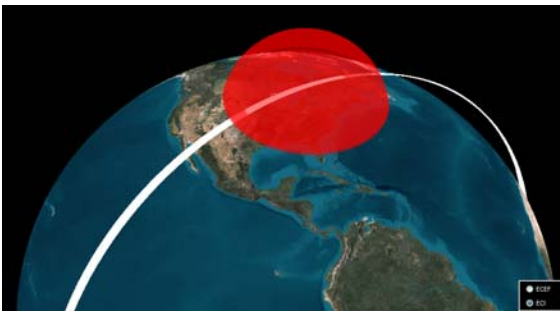


Figure 37: Ground Site FOV

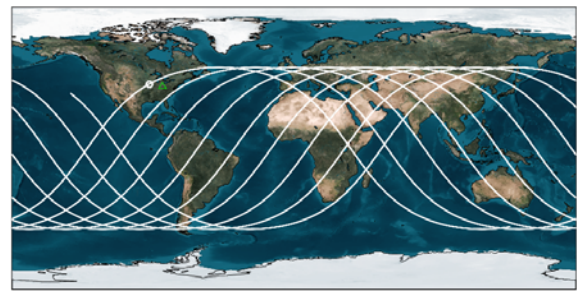


Figure 38: ISS Ground Track

A rare clear night in Dayton coupled with a long duration pass of the ISS presented a great opportunity to test the software on real-world observations. Tracking data is generated from a TLE file of 132 of the brightest objects and down selected for satellites which pass over the AFIT ground site at an elevation greater than 10 degrees

and a maximum visual magnitude during the pass of at least 5.0 between sunset of 5 January, 2016 and sunrise of 5 January, 2016. Of 132 objects, 41 satellites met this criterion. The telescope is first calibrated by pointing to level true north, or zero degrees azimuth and zero degrees elevation. The Meade telescope has a built in protocol for calibrating to level north by manual pointing to reference stars.

The open-loop tracking consists of pointing the telescope toward the azimuth and elevation of a propagated TLE position. Without feedback, the accuracy of the track is dependent on the accuracy of the TLE, the calibration of the telescope, and accuracy of the propagation. Video footage of the passes are captured with the StellaCam III and downloaded using the Epiphan software. An image of TeleTrak_v8 operating during the ISS pass with the target in the spotting scope FOV is shown in Figure 39.

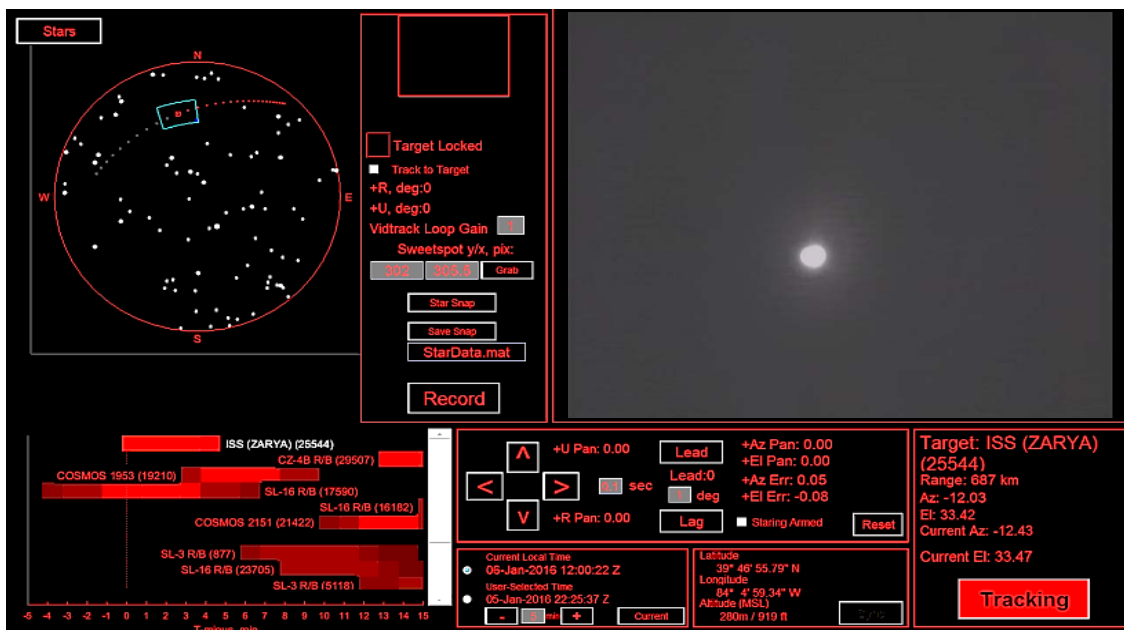


Figure 39: TeleTrak_v8 during ISS Pass, 6 January, 2016

Multiple targets were observed during observations. The success criterion for the open-loop tracking is acquiring the target in the spotting scope FOV such that a controller can drive the target to the main optic FOV. As shown in Figure 40, the spotting scope FOV is $1.62^\circ \times 1.22^\circ$ and the 10'' Meade main optic is $0.45^\circ \times 0.36^\circ$.

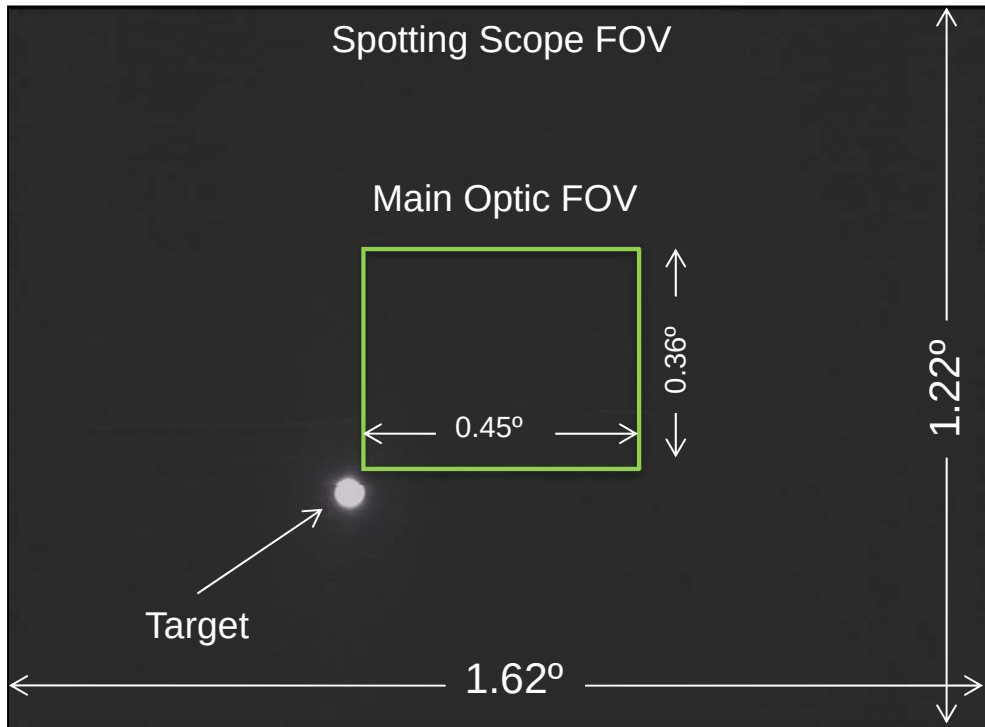


Figure 40: Orion Spotting Scope and Meade Main Optic Field of View

While acquiring the target in the spotting scope is considered successful, the performance of the open-loop tracking from a TLE file can be assessed by its ability to maintain the target in the main optic field of view. Manual controls are available to the user to pan around the target and lead or lag the target in its orbital path. The position of the target in the field of view was consistent throughout each individual track without user input; therefore, assessing the performance is based on whether the position of the target is passively held within the main optic throughout the entire track. An image

from the spotting scope for each target is shown in Figures 41 through 48 below compared to a green window representing the FOV of the main optic. Results are summarized in Table 4.

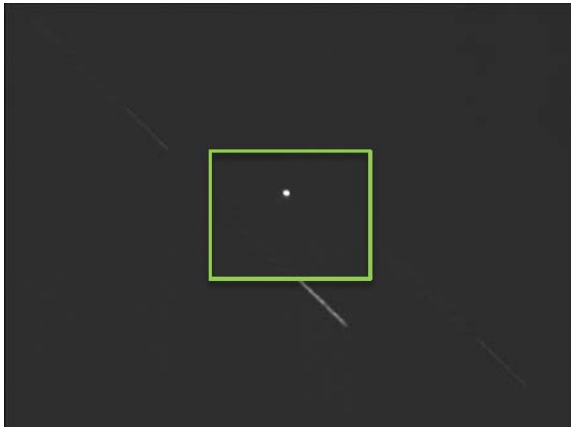


Figure 41: COSMOS 1544

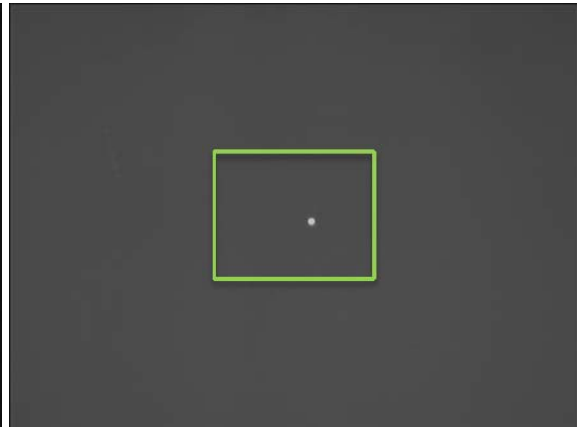


Figure 42: SL-16 R/B

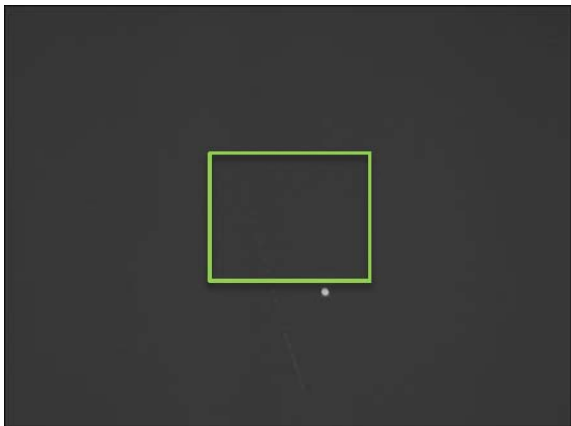


Figure 43: COSMOS 2151

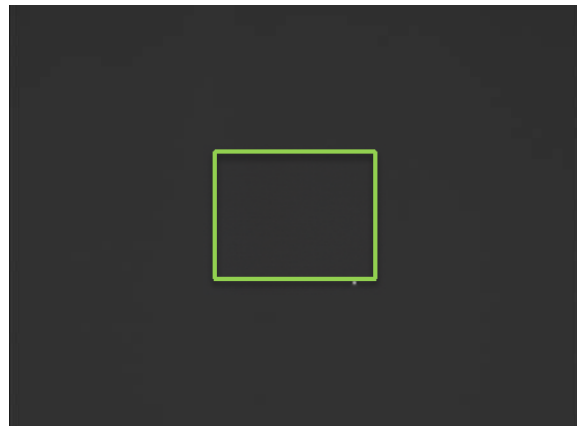


Figure 44: CZ4B R/B

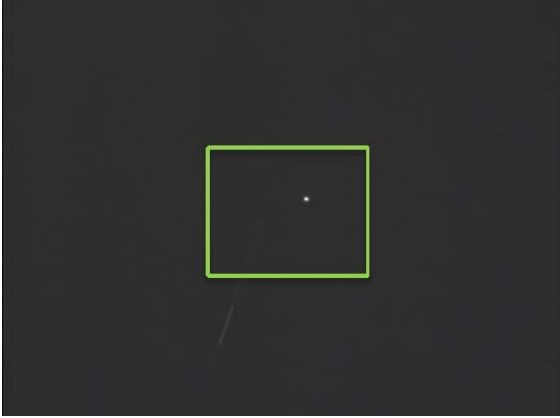


Figure 45: RESURS-DK 1

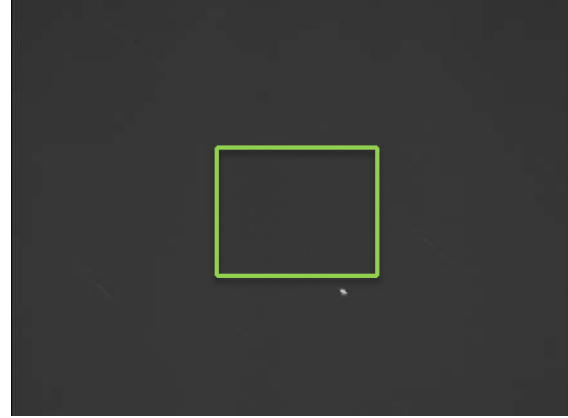


Figure 46: SL-3 R/B

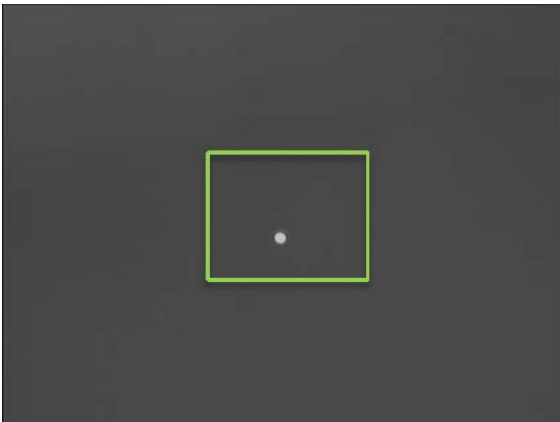


Figure 47: ISS Low Elevation

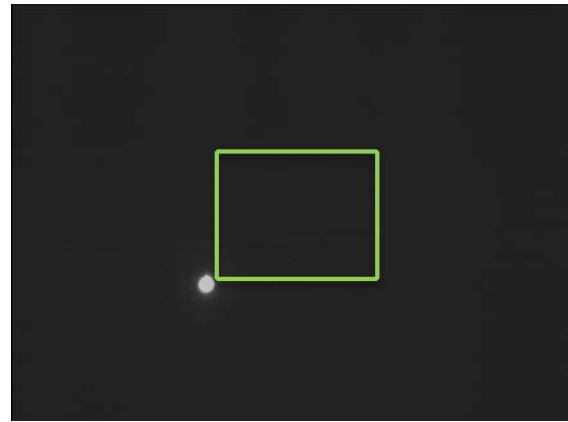


Figure 48: ISS High Elevation

Table 4: Success Rate of Observations

Target	Target in Spotting Scope FOV	Target in Main Optic FOV
COSMOS 1544	Success	Success
SL-16 R/B	Success	Success
COSMOS 2151	Success	Failure
CZ4B R/B	Success	Failure
RESURS-DK 1	Success	Success
SL-3 R/B	Success	Failure
ISS (ZARYA)	Success	Marginal Success

The results of the tracking show a reliable success rate for acquiring the target in the spotting scope based on the TLE. While the results show that the telescope is capable of predicting a target position and holding it in frame, the accuracy of the track is still dependent upon the accuracy of the TLE. The performance of the tracking shows a 57% (4 out of 7 targets) success rate of capturing a target in the main optic field of view. In the case of the ISS, the target is held in the main optic field of view for 50% of the pass, while the COSMOS 1544, SL-16 R/B, and RESURS-DK 1 targets are held in the main optic field of view for the entirety of the pass. These results are consistent with previous research which was the impetus for Gresham's work. For an open-loop track reliant on the accuracy of a TLE for a fast moving LEO object, these results are promising, but implementing the closed-loop control system is necessary for applications requiring reliable, high precision, or small field of view observations.

Hardware Independence

Gaining hardware independence is one of the primary objectives of this research motivated by the loss of tracking capability due to persistent operating system updates and changes to hardware components. Achieving independence from a specific operating system or coding language tool makes the system more robust and closer to reaching reliable operation with a small portable system. A large step toward independence is achieved by compiling the software tools into standalone executable files. Using MATLAB's deploy tool, programs can be compiled into a ".exe" file that can run on the same operating system type that it is compiled in, i.e. any Windows or Mac computer, without requiring MATLAB. The TeleCalc, TeleMap, and TeleTrak_v8

tools are successfully compiled and tested on multiple computers. The TeleTrak_v8 telescope commanding tool operates independent of MATLAB but tracking tests are conducted using the MATLAB script. Now that initial tracking functionality has been demonstrated, future work will be to test tracking from a dedicated computer independent of MATLAB.

One of the secondary objectives for this research was to regain the previous closed-loop tracking capability and performance of greater than 90% target time spent within a 0.05° FOV demonstrated by Gresham [8]. While closed-loop tracking was not tested, a proof-of-concept was demonstrated to acquire optical feedback independent of the camera and video capture hardware. Dependence on unreliable and fluctuating video hardware and drivers constrains the system to outdated and unsupported equipment. A more flexible approach is to capture streamed video data directly from the monitor which can operate with any camera and video streaming software. An optional screen capture function is integrated into the TeleTrak_v8 tool which captures a specified area of pixels from the monitor. When streaming video on screen from the Epiphan Frame Grabber or any commercial camera software, the screen capture function acquires the arbitrarily set 800x600 pixels and directs them to the TeleTrak_v8 interface as shown in Figure 49.

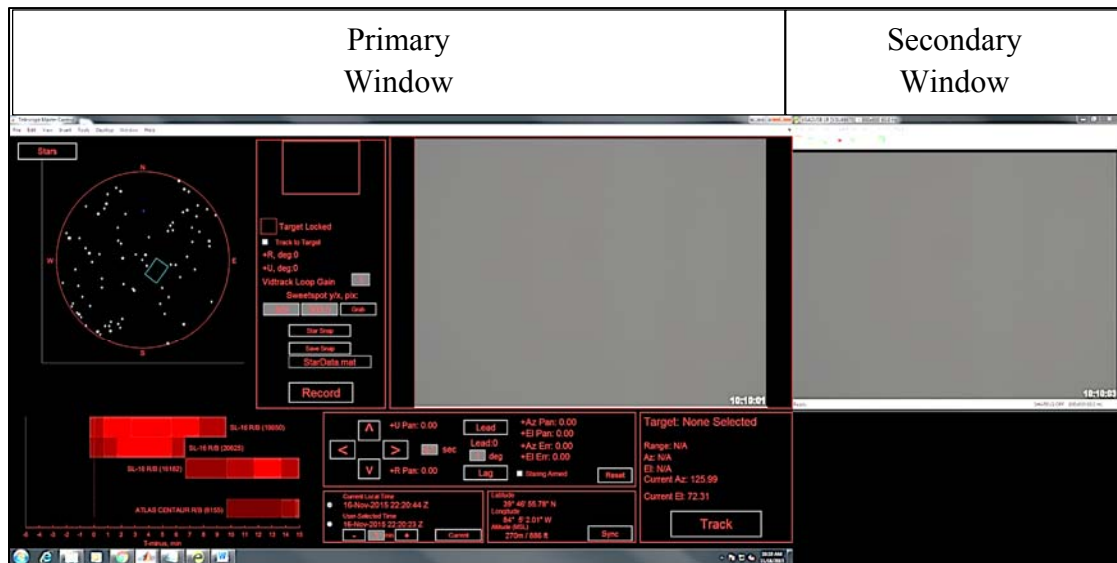


Figure 49: Screen Capture Technique for Optical Feedback

The screen capture technique is agnostic to device drivers, camera equipment, and telescope hardware. While the pixels are captured in real time, they can be analyzed using the same feedback control algorithm designed by Gresham. The current challenge of the screen capture approach is latency on the order of 1-2 seconds due to how the pixels are handled in the background. Currently, pixels are captured as an 800x600x3 8-bit unsigned integer (uint8) matrix with full RGB (0 – 255) values. This can be converted to an 800x600 uint8 matrix of grey scale (0 – 255) values, but the pixels should be converted during the capture routine before they are stored, not after. Although the specific processing constraints Gresham encountered when designing and testing the closed-loop controller are unknown, he identified a latency constraint due to the Meade serial interface. The mount firmware is limited to communicating azimuth and elevation position feedback approximately every 0.7 seconds to the external computer, and inbound data is delayed to the mount by approximately 0.26 seconds [8].

Optimizing the screen capture and pixel analysis process such that the latency is less than 0.7 seconds is necessary for the active feedback control system. While the 1-2 second latency is experienced on an (admittedly) low performance desktop computer, the controller has not yet been implemented which will require more processing power from an already saturated system. Furthermore, a future objective of TeleTrak research is to operate the system from a small automated computer, so processing power may become a consideration. This constitutes a first attempt at the screen capture technique and there is still a lot of room for improvement in the coding. Although the control system still needs to be implemented and tested, achieving active optical feedback control with a screen capture technique will be a large step towards hardware independence.

Summary

This chapter presented a comparison of the TeleCalc and STK orbit propagation models and results from the demonstration of the TeleTrak software tools for satellite observations. The resulting software tools developed and adapted during this research were compiled into standalone executable programs, enabling operating system flexibility and a simple method of sharing. A method for additional hardware flexibility is also presented by implementing a screen capture technique to retrieve optical feedback from the telescope camera.

V. Conclusions

Overview

This research set out to answer the following question. How to best reengineer existing code and develop a set of modularized software tools for orbit propagating, mission planning, and satellite tracking with COTS telescopes that are hardware agnostic and adaptable to future operating systems. This question was broken down into the following objectives and the results are given in Table 5.

Primary Objectives:

1. Regain previous open-loop tracking functionality of maintaining a target within the spotting scope FOV for the duration of a pass.
2. Establish a modular software architecture.
3. Compile tracking utilities into standalone executable programs for Windows and MAC operating systems.
4. Develop an orbit mapping and observation mission planning tool.

Secondary Objectives:

5. Regain previous closed-loop tracking functionality of greater than 90% target time spent within a 0.05° FOV.
6. Demonstrate desired tracking functionality on an Alt-Az and Equatorial telescope mount.

Table 5: Research Objectives Met

Objective	Met	Partial	Not Met
1	x		
2		x	
3	x		
4	x		
5			x
6		x	

Conclusions of Test Results (Objective 1)

Chapter 4 presented a comparison of the STK SGP4 propagation model to the MATLAB SGP4 model developed by Vallado, adapted by Beck, and vectorized by Schmunk. This comparison showed a substantial disparity in the initial and predicted location of the ISS from Orbit Epoch and these results were shown to have a significant potential impact on the telescope's capability to acquire the target in the spotting scope field of view. Still, the ISS and other targets were all successfully captured in the spotting scope field of view during observations from look angles generated by the MATLAB SGP4 propagator used in TeleCalc. 100% of targets were acquired in the spotting scope field of view and 57% of targets acquired within the main optic field of view. Results from telescope observations are the closest data available to determine "truth" of the target's position; however, observation data is still limited by errors in telescope calibration, motor control, and atmospheric effects. Although tracking was demonstrated in open-loop, the system's performance in maintaining the target within the main optic field of view four out of seven observations, and 100% of targets within the spotting scope field of view, validates both the TeleCalc propagation model and the Meade telescope's ability to execute commanded look angles. Chapter 4 discussed that a $1.62^\circ \times 1.22^\circ$ field of view camera would require an orbit prediction accuracy of $0.81^\circ \times 0.61^\circ$ assuming a perfect pointing accuracy. Using the ISS pass on the morning of 6 January, 2016 as a test case, The TeleCalc SGP4 to STK SGP4 model comparison concluded that the error between the two models exceeds the field of view threshold of the camera. Observations were taken using the TeleCalc model results; therefore, if the STK model results are taken as truth, the ISS would not have appeared in the camera

field of view for the first 1 minute and 20 seconds. The observational test data showed the ISS within the spotting scope field of view for the entirety of the track. Although the telescope pointing accuracy is not characterized in this research, the performance of the target tracking has been demonstrated to be consistent across multiple LEO passes, so it can be concluded that the TeleCalc SGP4 model predictions are closer to “truth” than the STK prediction from a given TLE file. Additional observations are required to characterize the accuracy of the two models; however, the TeleCalc model has demonstrated sufficient accuracy to acquire a target in the spotting scope field of view in order to be driven to the main optic field of view by a closed-loop tracking system, thereby meeting the research objective.

Software Architecture (Objective 2)

The objective to design and establish a modularized software architecture is partially met. The resulting architecture of TeleTrak exhibits many qualities of modularity and reusability, but the TeleTrak_v8 tool does not. The tools and services developed for TeleTrak and previously expanded TeleTrak applications are distinctly separated to support reusability; however, the TeleTrak_v8 Adapter Layer tool is constrained by internal string callbacks and should be handled as a tracking software tool unique to TeleTrak.

This research established a hierarchical component design for the TeleTrak software architecture and presented the methods for exploiting the modularity of the system. Through research and implementation, it was discovered that software architectures are inherently hierarchical. It is the execution of the design (how the

developer interacts with the architecture) that is the key to exploiting the hierarchical characteristics. Meeting this objective was a natural prerequisite to regaining operability. Familiarizing with and organizing the code into a common repository was necessary before beginning to identify the bugs and constraining components such as device driver specific references. Through implementation of the HCD architecture, the TeleCalc services are successfully distinguished and organized into modules as illustrated in Figure 26. Decomposition is essential to enabling the developer to exploit the separation of components which is reflected in the code through distinct callbacks and local variables. Minimizing dependencies was the most critical factor in achieving modularity, but the extent of which modularity could be achieved was limited. The callback functions of the TeleTrak_v8 tools are fixed to the Adapter Layer through callback strings and therefore do not express modularity. Callback strings handle all variables globally and prevent breakpoints for testing and debugging. If each function is written into discrete files, then all variables would need to be handled locally and retested. It was determined that the tracking tool would need to be deconstructed and rebuilt from the ground up, so instead the tool was adapted for new operating systems to regain initial operability and to enable a standalone compiled version.

Compile Tracking Utilities (Objective 3)

The TeleCalc, TeleMap, and TeleTrak_v8 software tools developed and adapted during this research were successfully compiled into standalone executable programs using the MATLAB deploy tool. This enables operating system flexibility and a simple method of sharing the TeleTrak tools. The Telescope commanding GUI, TeleTrak_v8,

is successfully compiled but remains untested as tracking operations during this research were conducted in MATLAB. A quick-start user guide and overview of the TeleCalc tool are presented in Appendices A and B. Appendix C outlines the process followed to compile the software packages using the MATLAB deploy tool.

Mission Planning and Orbit Mapping Tool (Objective 4)

One of the results of this research is the orbit mapping and mission planning tool TeleMap. Visualization is an essential component to mission planning and orbit analysis. Visualizations provide a broader context for the relationships between TLEs, Orbits, Ground Sites, and observation accesses. This tool has been powerful for planning observation opportunities and understanding the orbits of real world satellites. The program enables the user to establish ground sites, define their range and fields of view, and plot satellite positions at any point in time during the observation night. When plotting orbits in the 3D globe, plots can also be shown in the ECEF or ECI frames. For example, the operational GPS constellation is plotted in the ECEF frame in Figure 50, and the ECI frame in Figure 51.

One of the primary motivators for TeleTrak research is the congested, contested, and competitive nature of space. Illustrating the actual congestion of Earth's orbit is a powerful motivator for SSA research. For this reason, a Master TLE file of approximately 15,400 tracked objects is downloaded and their location on 27 January, 2016 at 10:50:00 UTC is plotted in Figure 52, Figure 53, and Figure 54.

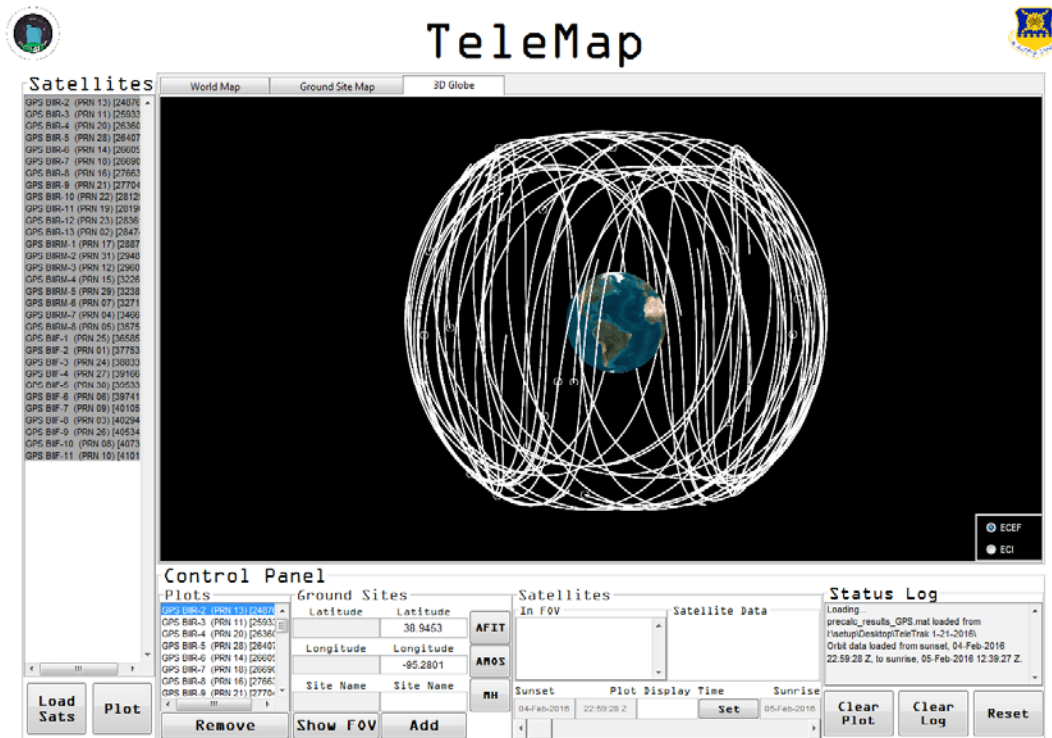


Figure 50: Operational GPS Constellation in ECEF Frame

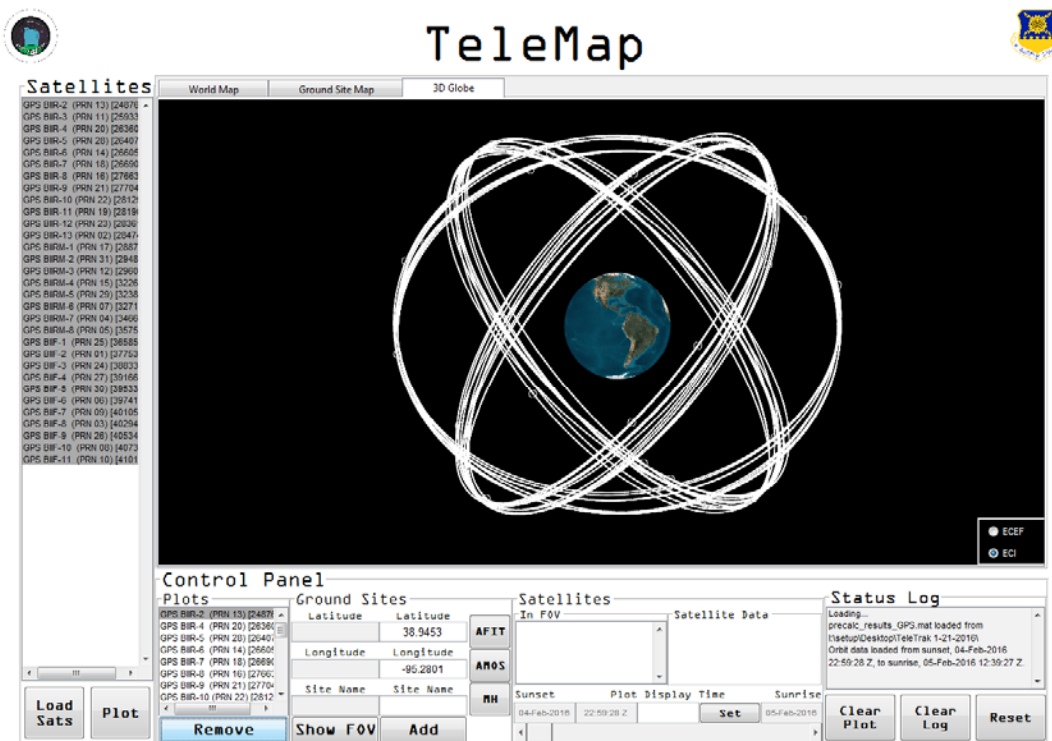


Figure 51: Operational GPS Constellation in ECI Frame

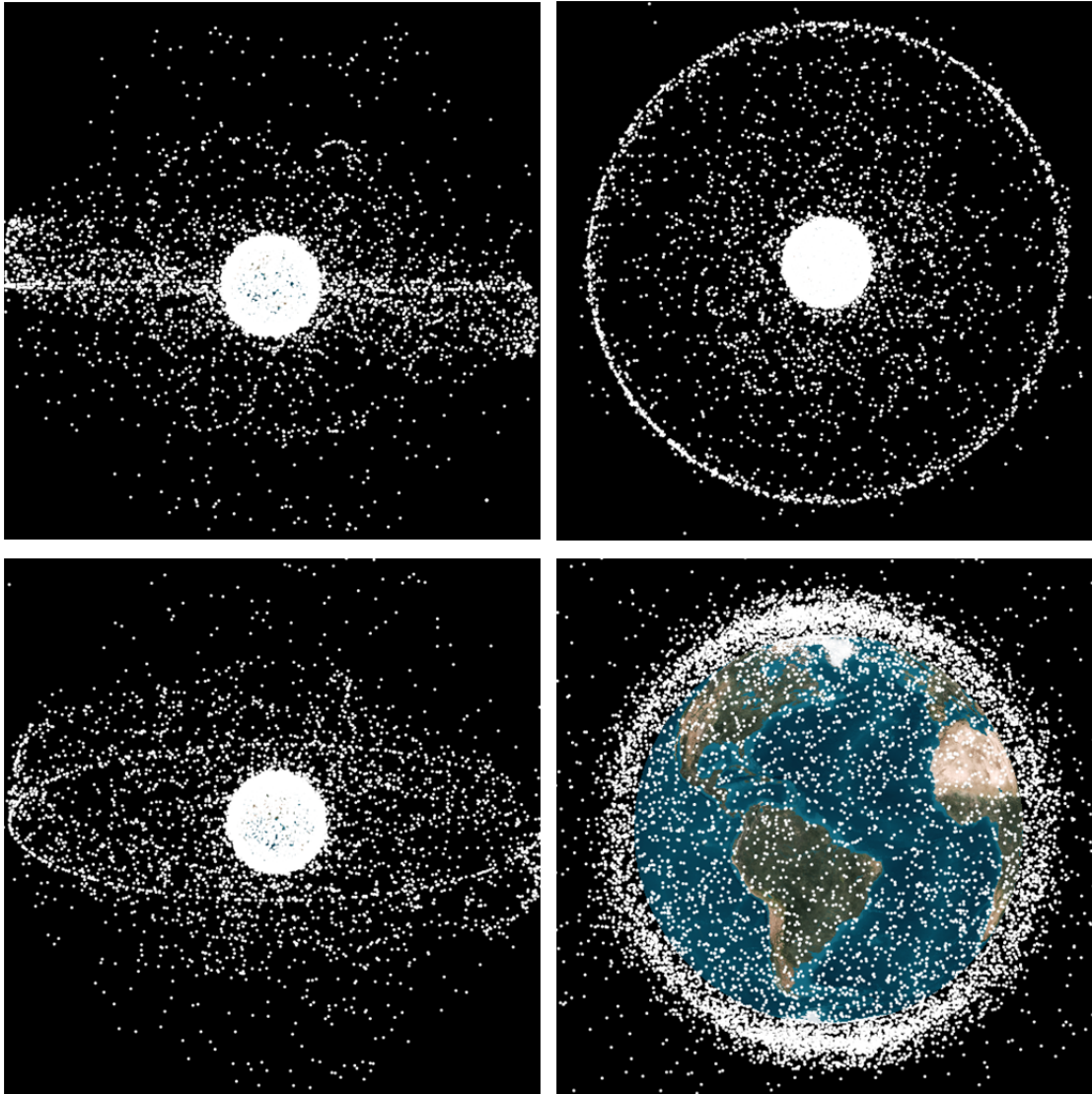


Figure 52: Full TLE Catalog Plots

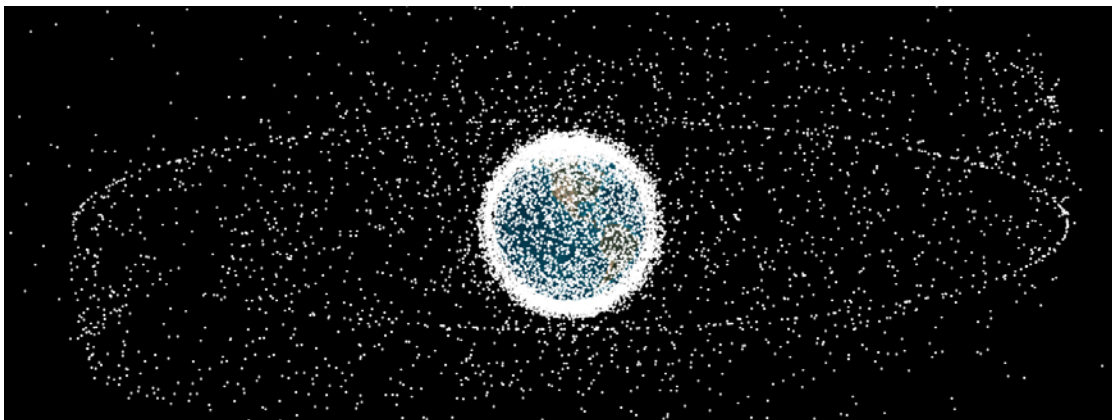


Figure 53: Full TLE Catalog Plot, GEO Regime

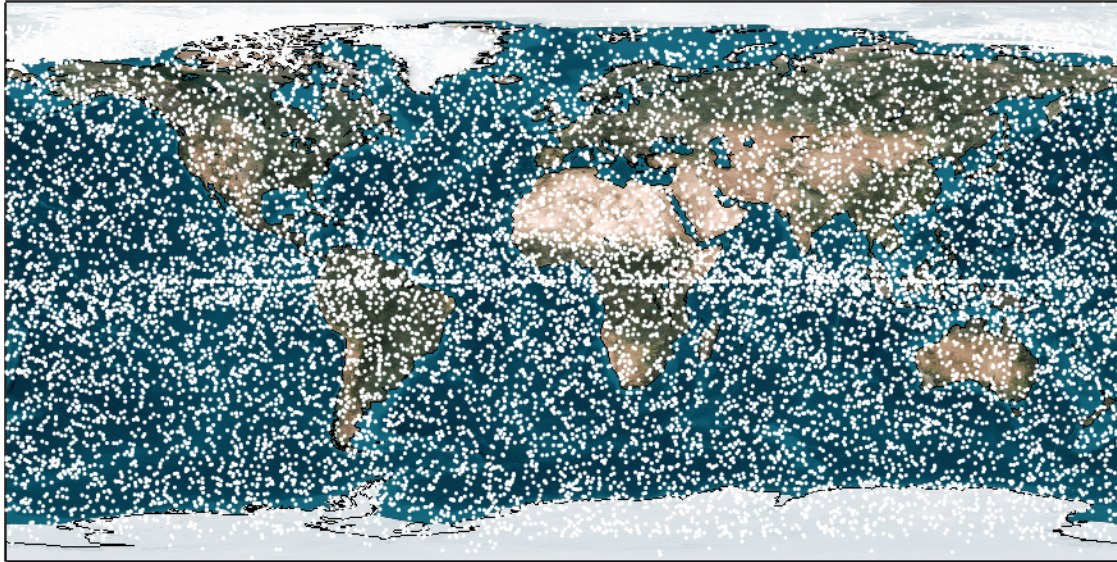


Figure 54: Full TLE Catalog Ground Track

Integrate Closed Loop Tracking Functionality (Objective 5)

Although unsuccessful in regaining closed-loop tracking functionality, this research did present a generalized method for implementing feedback using the screen capture technique discussed under future work. Minimizing dependency on device drivers and specific hardware is essential to ensuring the resilience of the system through the constant updates inherent of the computer industry.

Demonstrate Tracking on an Alt-Az and Equatorial Mount (Objective 6)

The testing results presented in this research were accomplished on an Alt-Az mount but operation from an Equatorial Mount is left for discussion under future work.

Significance of Research

This research has laid down a foundation designed to be built upon by future researchers in order to realize a significant cost saving technology using COTS

telescopes to enhance SSA. Previous work at AFIT has demonstrated the potential significant contributions to SSA using the TeleTrak system; however, the rapid pace of commercial updates to software and operating systems has left this capability inoperable. Learning from this, software architecture was established and designed to be reusable, modular, and robust to future updates. This was done by organizing and identifying the key layers of the software structure for future developers, developing easy to use interfaces that confine the limitations of the system, and compiling the software tools into standalone executable programs to operate independent of a specific coding language or operating system. Organizing the software source code and compiling TeleCalc and TeleMap into executable programs ensures the tools are shareable and available in the future to all Department of Defense researchers interested in maturing the technology of object tracking and identification with cost effective commercial telescopes in an operational environment.

Future Work

The two secondary objectives this research did not meet were to demonstrate previous closed-loop tracking functionality of greater than 90% target time spent within a 0.05° FOV as well as to perform tracking functionality on both an Alt-Az and Equatorial mount. The previously designed closed-loop tracking controller demonstrated promising results, but this capability was lost due to dependency on specific hardware and software. Methodology for implementing a screen capture technique is presented which removes the dependency on commercial hardware and camera device drivers, greatly increasing the flexibility of the system. The current

challenge of this method is the latency and processing power required to capture and handle each pixel. A method of efficiently converting the RGB values on the screen to grey values, or simply capturing a smaller number of pixels, will be required to minimize the latency such that the feedback can be consistently provided to the controller at a rate less than 0.7 seconds, the rate at which Gresham identified the Meade telescope's internal communication system was updating its position to the computer [8]. Gresham's analysis of the video stream at a rate of 15 Hz may be viewed as a benchmark to recreate the performance of his control system.

The telescope was operated from an Alt-Az mount during this research. AFIT currently has a Meade telescope on an Equatorial mount which can be tested to demonstrate increased hardware flexibility. The advantage of the Equatorial mount is that its axis of rotation is aligned with the Earth's axis of rotation, simplifying the tracking of celestial objects which are fixed in the night sky. This has obvious advantages for observing Geostationary or slow moving objects. An option was integrated into TeleCalc allowing the user to select the type of mount for which the calculations should be performed. This option was removed because it limits the flexibility of the system after the calculations are performed. Designing a translator which determines the type of mount the computer is connected to and translates the commands between an Equatorial and Alt-Az frame is the recommended approach for future development.

The three Adapter Layer tools TeleCalc, TeleMap, and TeleTrak_v8 are all functional works in progress. Their performance has been demonstrated during this research, but there is near limitless room for improvement. The TeleCalc tool relies on

text files containing sun vectors and satellite visual magnitude data that quickly become outdated and require updating. Developing a method of automatic updates or generating this data internally rather than ingesting data from files would be a big step in improving system longevity. The TeleMap tool is immensely useful for visualizations, orbit mapping, and mission planning, but realizing the full potential of such a utility requires greater development time. It is the desire of the author that students continue to build more functionality into TeleMap such as attributing an internal propagation tool so orbits can be plotted from any start and stop time independent of TeleCalc. The TeleTrak_v8 GUI has a lot of potential for greater levels of autonomy. Designing a smart system to plan and command the observations of a series of targets would be a large step towards a fully autonomous tracking system. TeleTrak_v8 is successfully compiled into a standalone program but remains untested during observations. Verification tests would finalize the independence from MATLAB and potentially enable tracking from a small dedicated computer. Previous work has identified a wide variety of TeleTrak applications. Maximizing the autonomy of the system will continue to demonstrate innovative approaches to applying cost effective contributions to SSA.

Lessons Learned

Reengineering existing code without a set foundation or Definition Layer is considerably more challenging than building modular code from the ground up. When reusing software, the architecture, whether established or not, and dependencies built into the software are all inherited and difficult to change without complete redevelopment. A documented definitions library, configuration control scheme, and

architecture must exist to build from; otherwise, a great deal of time is spent learning the software's definitions, services, and applications. This could lead to a mindset of "if this is how it was done, it must be right", which leads to overlooking errors and potential flaws in the software. When a set of software tools is not reusable and restricted to its original application, it risks becoming obsolete with the steady update of operating systems and hardware tools, as was the case with TeleTrak. Following the principles of software reusability becomes paramount for increasingly complex systems and can save significant time and money.

Software architectures are conceptual but reflected in the design and division of the services. Much of the modularity built into TeleTrak during this research relies on the divisions of functions and the separation of software services. While building a reusable system requires judicious design principles, the emergent aspects of modularity are conceptual, and it is incumbent upon the developer to familiarize with the software structure in order to exploit the modularity. The hierarchy component design provided a simple and understandable approach to constructing the architecture of an existing set of software, although there is certainly more room for depth. With the knowledge that most TeleTrak developers, including the author, are not software engineers, a simplified approach is preferred for users to get up to speed quickly. The ultimate goal of the framework and organization is to maximize reusability and rapid development from new students and researchers. The hierarchy component design met this objective well.

Software is just as fragile as hardware. A great deal of time was spent investigating errors with no explanation. In one example, the culprit was due to the

extrapolation of star pass arcs. The star propagations relied on having at least two points above the horizon to compute azimuth and elevation rates; however, on one rare occasion a star only appeared above the horizon for a single data point. This was never foreseen, and a lot of digging was required to identify the illusive error and modify the code to ignore single point star passes. This is one of many examples of unpredictable errors that are encountered in a relatively new set of software. Even in cases of software suites that have been continually tested and verified over many years, entire teams of software engineers are required for supporting updates and addressing errors. The lesson learned here is recognizing the fragility of software and the increasing necessity for simplified organizational architectures for increasingly complex software systems.

Summary

This research has addressed methods and principles critical for all engineers to practice when developing software that contributes to the Air Force mission. Software has clearly defined itself as the key to future technologies, and understanding and exploiting the cost saving and technological potential of smart, reusable software design is paramount for the Air Force to maintain dominance in the operational domain.

Appendix A – TeleTrak Quickstart Guide

This guide outlines 4 steps to using the TeleTrak software for satellite tracking.

Step 1: Acquire a TLE file

As of February 2016, NORAD updated TLE files are available from either celestrak.com without an account or space-track.org with an account. TLE files can either be saved as a “.txt” or “.tle” file type. Note: The TLE must be in the three-line format shown below including the satellite name.

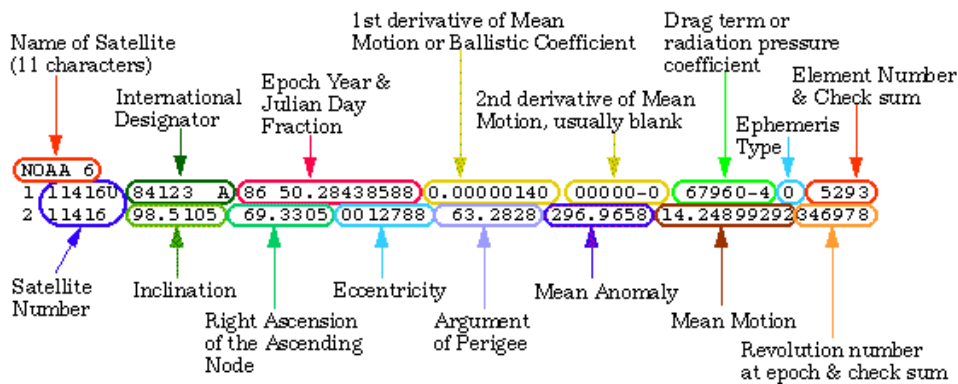


Figure A 1: Two-Line Element Set Definition [21]

Step 2: Run TeleCalc

[TeleCalc.m – MATLAB 2012a or higher / TeleCalc.exe – Any Windows machine]

The TeleCalc GUI provides an interface for the user to customize the satellite down selection criteria. Input boundary conditions and parameter descriptions are presented if the user holds the mouse over the input parameter. If a user wants to create orbit mapping data for a night of observations that includes all objects in the TLE file, they can set the input criteria to include all visual magnitudes by entering “N/A” for Minimum Satellite Brightness Threshold and include all orbital periods. If the user selects “Run TeleCalcs” and all input criteria are within bounds, the user will be

prompted to select the desired TLE file. After calculations are performed, the user will be prompted to name the “*.mat” output file and location. A “Help” pop-up box is accessible on the bottom left of the GUI.

Step 3: Run TeleMap (optional)

[TeleMap.m – MATLAB 2015a or higher / TeleMap.exe – Any Windows machine]

The TeleMap GUI is not required for tracking but plots satellite positions propagated from the TeleCalc tool throughout a night of observations for visualization and mission planning. First select “Load Sats” in the lower right to select the desired “*.mat”

TeleCalc output file. The satellites in the output file will be displayed in the left “Satellites” list-box and the sunrise and sunset times that encapsulate the object’s propagation appear in the “Satellites” panel with a push button and slider to specify the

time at which the satellite’s predicted position will be plotted within the timespan. By highlighting satellites in the “Satellites” list-box and selecting “Plot”, predicted orbits and positions will be plotted in the World Map, Ground Site Map, and 3D Globe tabs.

The 3D Globe can be rotated and interacted with using the MATLAB figure tools. A radio-button panel in the lower right of the 3D Globe tab allows the user to display 3D plots in either the ECEF or ECI frame. The “Ground Sites” panel allows users to add ground sites anywhere on Earth by latitude and longitude and display their field of view by highlighting them from the “Plots” list-box and selecting “Show FOV”. Plotted satellites and ground sites will appear in the “Plots” list-box and can be removed by highlighting them and selecting “Remove”.

Step 4: Run TeleTrak_v8

[TeleTrak_v8.m – MATLAB 2012a and higher / TeleTrak_v8.exe – Any Windows Machine]

The TeleTrak_v8 GUI is the interface for executing tracks and commanding the telescope. The user is first prompted with four inputs:

Input "0" to start at computer time, "1" to simulate from precalc_results start time [0]:

Input "0" if you have COM ports or USB-to-Serial, "1" if you don't [0]:

Input "0" for color output, "1" for red-and-white binary [0]:

Input "1" if the Telescope Camera is streaming in the designated monitor space [0]:

Executing the “Enter” key for each input defaults to tracking operations based on the computer’s clock using serial COM ports for telescope communication, color output, and without screen capture for optical feedback. Once entered, the user is prompted to select the “*.mat” file that includes the tracking data. The software will search the computer’s serial COM ports for a Meade telescope and, whether found or not, deploy the GUI. Searching for camera devices is currently suppressed and should be handled outside of the GUI for a more device independent approach. As targets appear in the scheduler in the bottom left they can be selected and their predicted pass will be plotted on the star map. Before or during a pass, the “Track” button can be selected and the telescope will point to the target’s propagated position or anticipate the pass by pointing to the target’s predicted azimuth at minimum elevation.

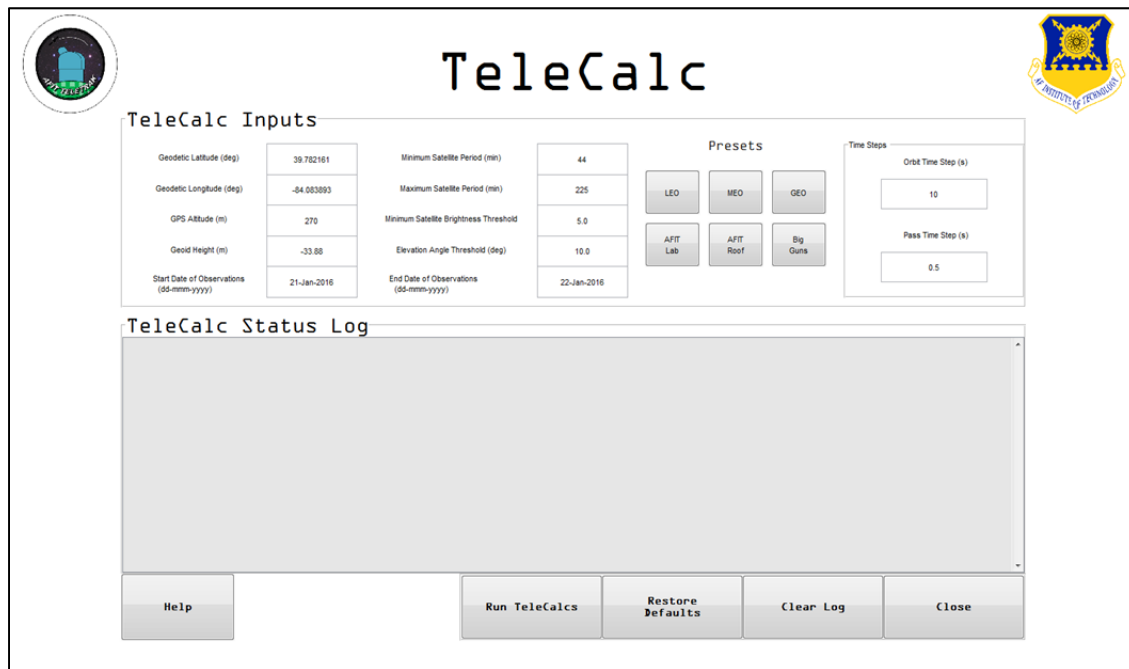
Appendix B – TeleCalc Overview

The purpose of this Appendix is to provide an overview of the behind-the-scenes calculations performed by TeleCalc to produce the output tracking data file.

1. TeleCalc User Input Menu

- a. This Graphical User Interface (GUI) provides the user with the following list of inputs that are set to default settings (LEO observations that night from LEO) and accepts appropriate ranges of values.
- b. Once “Run TeleCalcs” is selected, the user is prompted to locate the latest TLE file and all user inputs are forwarded to watcher_precalcs.m.

The status of the calculation process is displayed in the “TeleCalc Status Log”.



The screenshot displays the TeleCalc graphical user interface. At the top center is the title "TeleCalc". On the left is a circular logo, and on the right is a shield-shaped logo. The main area is divided into several sections:

- TeleCalc Inputs:** A table of input fields with the following values:

Input	Value
Geodetic Latitude (deg)	39.762161
Geodetic Longitude (deg)	-84.083893
GPS Altitude (m)	270
Geoid Height (m)	-33.88
Start Date of Observations (dd-mm-yyyy)	21-Jan-2016
End Date of Observations (dd-mm-yyyy)	22-Jan-2016
Minimum Satellite Period (min)	44
Maximum Satellite Period (min)	225
Minimum Satellite Brightness Threshold	5.0
Elevation Angle Threshold (deg)	10.0
- Presets:** A set of buttons for selecting observation types: LEO, MEO, GEO, APIT Lab, APIT Roof, and Big Guns.
- Time Steps:** Two input fields for "Orbit Time Step (s)" (set to 10) and "Pass Time Step (s)" (set to 0.5).
- TeleCalc Status Log:** A large, empty rectangular area for displaying the status of the calculation process.
- Bottom Buttons:** A row of five buttons: Help, Run TeleCalcs, Restore Defaults, Clear Log, and Close.

Figure B 1: TeleCalc Graphical User Interface

2. Compute Star Data

- a. First the sunset and sunrise times are calculated to the nearest minute for the observation night using [getdarkness.m](#). This is the interval during which the star positions will be computed.
- b. Star data is imported from the Yale Bright Star Catalog (BSC) using the [killer_getstars.m](#) function. The BSC file is titled “catalog.dat” and a second file is necessary containing nutation terms called “circ179_nutation_terms.mat”. [killer_getstars.m](#) produces azimuth and elevation data that agree within 1 arcmin (0.017 deg) of results produced by Cartes du Ciel v2.76.
- c. Splines are then computed for the selected stars. The Azimuth, Elevation, Range, Azimuth-rate, Elevation-rate, and Range-rate are all computed using the [spline.m](#) embedded MATLAB function. Note that this process DOES NOT account for multiple star “passes” during a single run. The script examines a single sunset-sunrise period in which this should never occur except at extreme latitudes.

3. Import and Categorize Satellite Data

- a. The function [getcatalogsats.m](#) is used to read from the selected TLE file and discards satellites that do not meet the input criteria for maximum and minimum period and visual magnitude are discarded. A period of 44 minutes to 225 minutes is considered a LEO satellite by the NORAD deep-space definition (SGP4 uses different calculations past this

regime). MEO is defined as 225 – 1300 minutes and GEO is defined as 1300 – 1800 minutes.

- b. Two common methods are used to catalog visual magnitude, the “McCant” and “Molzcan” methods respectively. The McCant method represents a best-case scenario. To not preclude potential viewings from the user, this process converts more conservative Molzcan numbers to a McCant equivalent value.
- c. Three different brightness catalogs are used to discard satellites that don’t meet the user selected brightness threshold. The first is the ‘[qs.mag](#)’ file which is distributed by Mike McCant defining standard brightness at 1000km range and full-phase (i.e. the angle between the observer-to-satellite-to- sun vector is 180 degrees). The second is the ‘[mcnames](#)’ file distributed by Ted Molzcan defining standard brightness as 1000km range and half-phase (i.e. the observer-satellite-sun angle is 90 degrees). When a visual magnitude is referenced from this source, its value is converted to a McCant equivalent value. The final source is ‘[ha_gps.txt](#)’ and comes from Heavens-Above.com. This is an older reference and only used if a satellite does not first appear in ‘[qs.mag](#)’ and ‘[mcnames](#)’ in that order. In the event a visual magnitude is not available in any of these three catalogs, the user is prompted to enter their own predicted visual magnitude.
- d. Satellites that meet the user input criteria are propagated using [sgp4_vectorized.m](#). This function is a vectorized version of [sgp4.m](#)

provided by AIAA 2006-6753 available at (<http://celestrak.com/publications/AIAA/2006-6753/>). The propagating procedure uses a combined version of SGP4 and SDP4 prediction models from Space Command. Position data including lat/long and x-y-z data in the ECEF and ECI frame is generated from the propagation at a user specified time step for plotting in TeleMap.

- e. Satellite orbits are also propagated using the same sgp4 model at a coarse time step of 1 minute as constrained by the TeleTrak_v8 scheduler update rate for determining ground site passes.

4. Get Site Data

- a. The function [getsite.m](#) is used to produce geocentric (ECI) position vectors for the user specified ground site location by converting from ECEF to ECI in a one-rotation transformation matrix.
- b. Geocentric (ECI) position vectors of the sun are computed using [getsun.m](#) and the 'sun.txt' file which provides physical properties and ephemeris data from 20 August, 2015 – 20 August, 2017.
- c. Next it is determined which satellites pass over the specified site above a user selected elevation threshold (defaulted at 10 degrees). Any unobservable satellites are marked with a maximum visual magnitude of 99.

5. Calculate Satellite Brightness

- a. For satellites that pass over the ground site with max brightness below the threshold, visual magnitudes are calculated for the duration of the

passes using the formula provided by Matson at

(<http://www.satobs.org/seesat/Apr-2001/0313.html>). Time during the pass when the satellite does not meet the user-selected visual magnitude threshold is again set to a magnitude of 99.

6. Generate Tracking Data From Down Selected Satellite Orbits

- a. The original list of satellites read from the selected TLE file has now been truncated by passes unique to the site location and estimated visual magnitudes. Now the orbits of these satellites are propagated for the duration of their visible time over the site at a time step of 0.5 seconds (This can be changed in the code).
- b. The functions [getsite.m](#) and [getsun.m](#) are used again to capture the site and sun positions at the finer time step. This time, [getLAST.m](#) is also used to return the angle between the positive x-axis (or vernal equinox) and the site defined by local GPS longitude.
- c. The function [sgp4_vectorized.m](#) is used again to propagate the selected satellite orbits and interpolation polynomials for azimuth, azimuth rate, elevation, elevation rate, range, range rate, and predicted brightness are produced at the finer time step and only during satellite visibility.
- d. Finally, satellite names, azimuth, elevation, rates, visual magnitudes, stars, and site location data from sunset to sunrise of the selected night for satellites that pass over the specified ground site are all saved to a [‘*.mat’](#) file with a user specified name.

Appendix C – MATLAB Deploy Tool

The purpose of this appendix is to outline the method used to compile executable programs with the built in MATLAB Deploy Tool. (Note: The deploy tool is available in versions of MATLAB as early as 2009, but the MATLAB 2015a procedure will be discussed here.) A video walkthrough is available on Mathworks at: (<http://www.mathworks.com/videos/getting-started-standalone-applications-using-matlab-compiler-100088.html>.)

To compile standalone applications in MATLAB 2015a, begin by selecting “Application Compiler” from the MATLAB APPS menu, or by entering “deploytool” in the MATLAB command window and selecting “Application Compiler”. The user will be presented with the compiler window below.

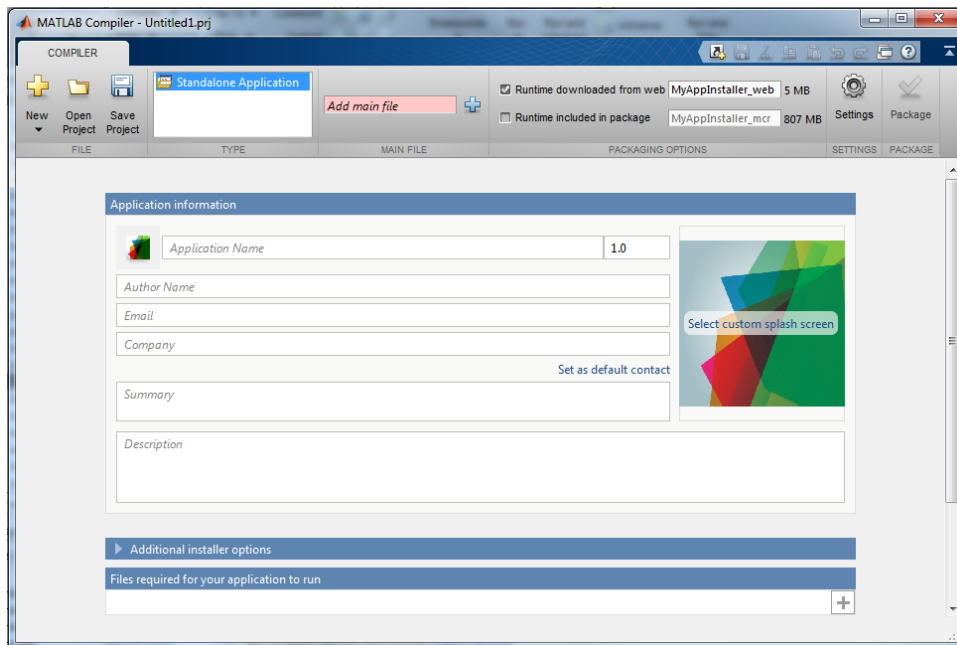


Figure C 1: MATLAB Compiler Window

Start by selecting “Add main file” at the top and selecting the central MATLAB GUI file. The files required to run the GUI will automatically be added to the “Files

required for your application to run” panel; however, MATLAB will not recognize data files or other inputs required of the application, so it is recommended to add each GUI dependent file manually. The “Packaging options” panel default minimizes the application file size by letting the final package download the MATLAB Runtime libraries from the web when the new user executes the program for the first time. By selecting “Runtime included in package”, the Runtime libraries will be included in the installation package. The final step is to select “Package” and the MATLAB compiler will generate the sharable “.exe” file for the program. When executed, the program will automatically install onto the computer and can be run like any other program.

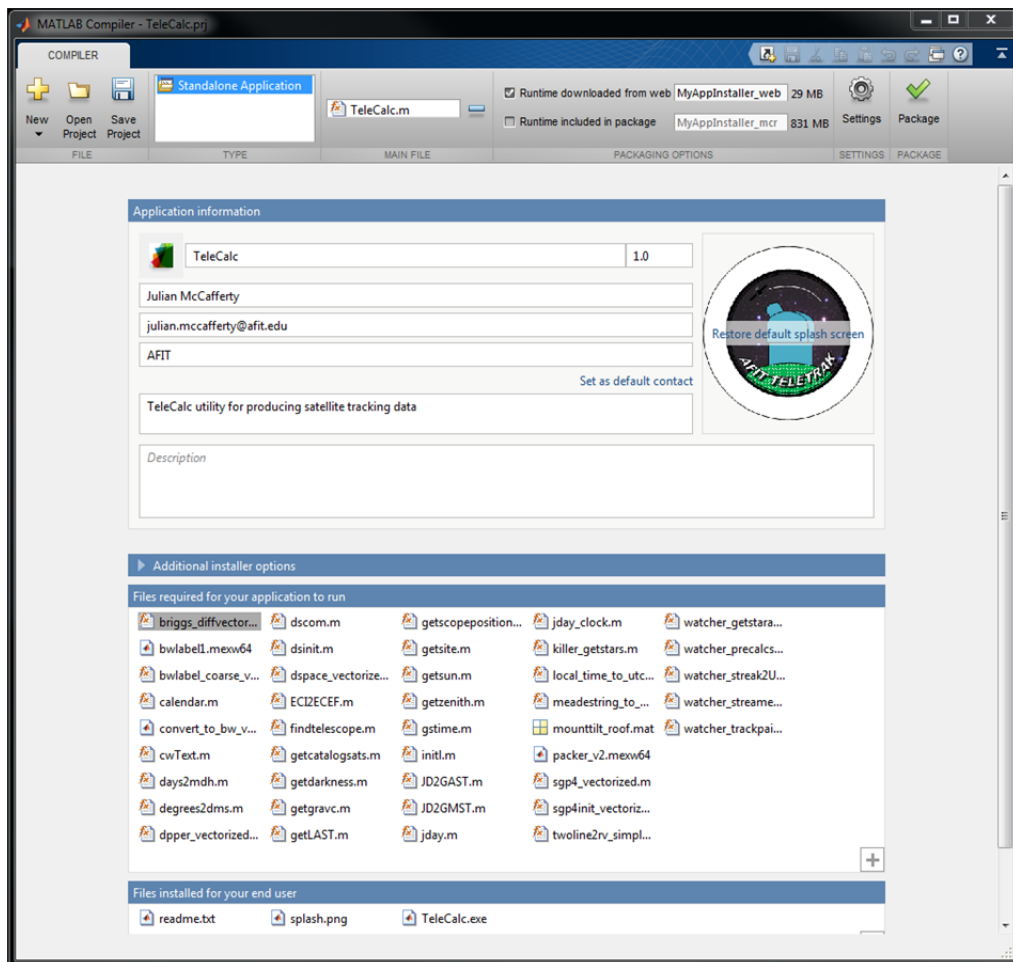


Figure C 2: MATLAB Compiler Example

Appendix D – Architecture and Design Documents

This appendix contains the architecture design documentation in dependency diagrams and tables defining the required reference files used by each application. The “*.mat” file in Figure D-1 is the TeleCalc output file named by the user containing the tracking data. Table D-3 defines the data structure of the “*.mat” TeleCalc output file referenced by TeleMap in Figure D-2 and TeleTrak_v8 in Figure D-3. The green functions in each dependency diagram are the integrators which reference the lower level services.

TeleCalc Architecture Design

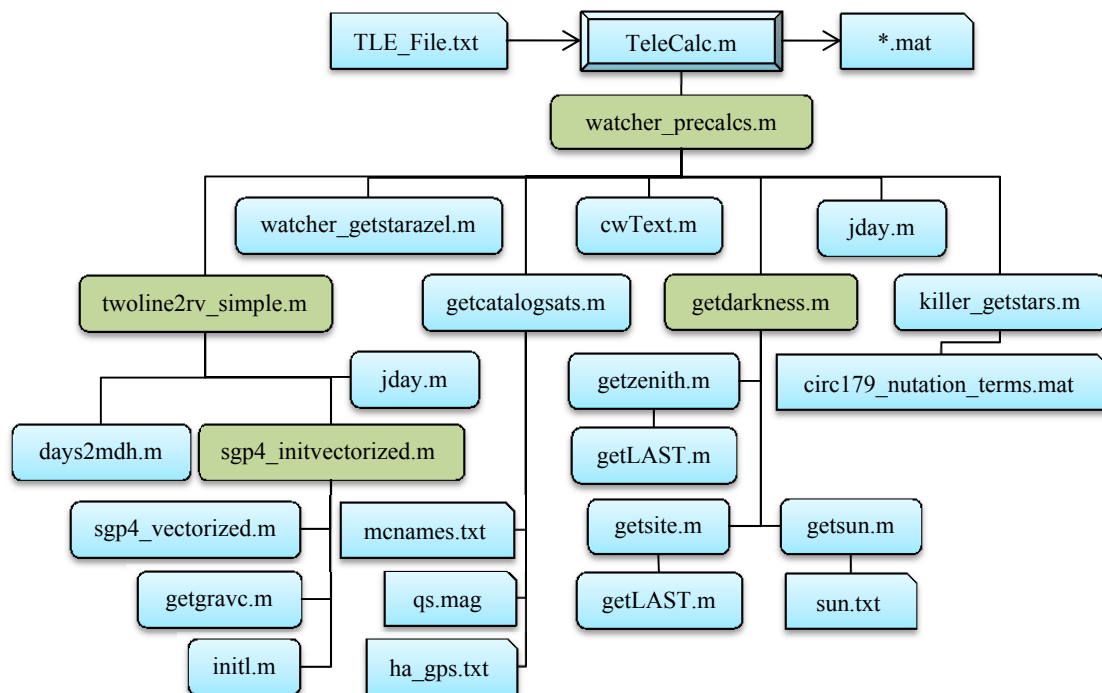


Figure D-1: TeleCalc Dependency Diagram

Table D-1: TeleCalc Reference Data File Descriptions

Reference File	Reference Service	Description
TLE File	watcher_precalcs.m	NORAD Two-Line Element Set Input File (*.txt or *.3le)
catalog.dat	killer_getstars.m	Yale Bright Star Catalog (BSC) file containing bright reference stars
circ179_nutation_terms.mat	killer_getstars.m	Earth nutation terms to increase star position accuracy
sun.txt	getsun.m	Sun ephemeris data from 2015-Aug-20 to 2017-Aug-20 (requires updating)
ha_gps.txt	getcatalogsats.m	Heavens Above brightness file of known satellite visual magnitude data
mcnames.txt	getcatalogsats.m	Molzcan brightness file of known satellite visual magnitude data
qs.mag	getcatalogsats.m	McCants brightness file of known satellite visual magnitude data

Table D-2: TeleCalc Output File Definitions

Variable	Description
databaseindex	Index of object ID values which meet user criteria
height_alt_gps	GPS altitude of user selected ground site location
height_geoid_gps	Geoid height of user selected ground site location
lat_local_gps	Latitude of user selected ground site
long_local_gps	Longitude of user selected ground site
passdatabase	Includes all object position and time data in SEZ frame for each object passes that meet user criteria between sunset and sunrise of the user selected observation night
r_ECEF_vec	Contains x,y,z position data (km) in ECEF frame and Julian Date time stamps for each object from sunset to sunrise of the user selected observation night (used for mapping)
r_ECI_vec	Contains x,y,z position data (km) in ECI frame and Julian Date time stamps for each object from sunset to sunrise of the user selected observation night (used for mapping)
Sat_lla	Contains object name, ID, Latitude (deg), Longitude (deg), and Altitude (m) between sunset to sunrise of the user selected observation night (used for mapping).
sat_splintype	Defines spline type used for interpolating target's position and rates (6-cell for satellites/arcs or 7-cell for stars/rasters)
satindex	Contains object ID, Name, Standard Magnitude, and source of Standard Magnitude data for objects that meet user criteria
sscindex	Contains object ID of all objects in TLE file
star_slinetype	Defines spline type used for interpolating target's position and rates (6-cell for satellites/arcs or 7-cell for stars/rasters)
stardatabase	Includes star position and time data in SEZ frame for bright stars that appear in the night sky between sunset and sunrise of user selected observation night
stardatabaseindex	Star ID numbers of all program selected bright stars
stars	Star ID number and brightness data for all stars in Yale Bright Star Catalog (BSC) considered for "stardatabase"
vmag	User Selected minimum visual magnitude threshold criteria used for down selecting from TLE file

TeleMap Architecture Design

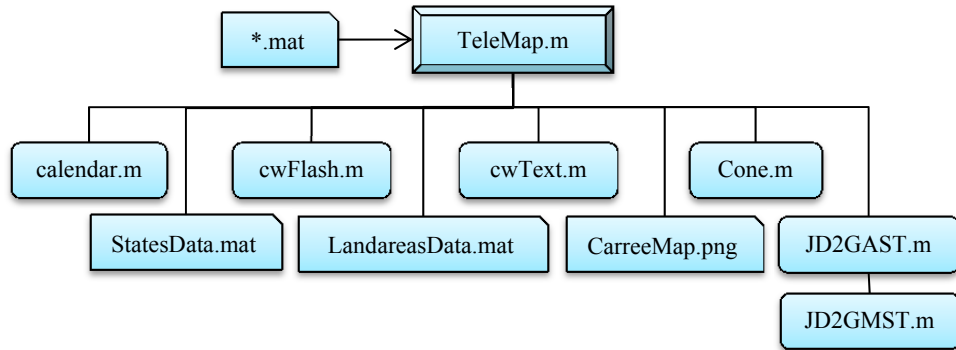


Figure D-2: TeleMap Dependency Diagram

Table D-3: TeleMap Reference Data File Descriptions

Reference File	Reference Service	Description
CarreeMap.png	TeleMap.m	Image file of Plate Carree projection for world map and 3D globe
LandareasData.mat	TeleMap.m	Matrix file of Earth land mass coast locations for world map
StatesData.mat	TeleMap.m	Matrix file of North American State borders for ground site map

TeleTrak_v8 Architecture Design

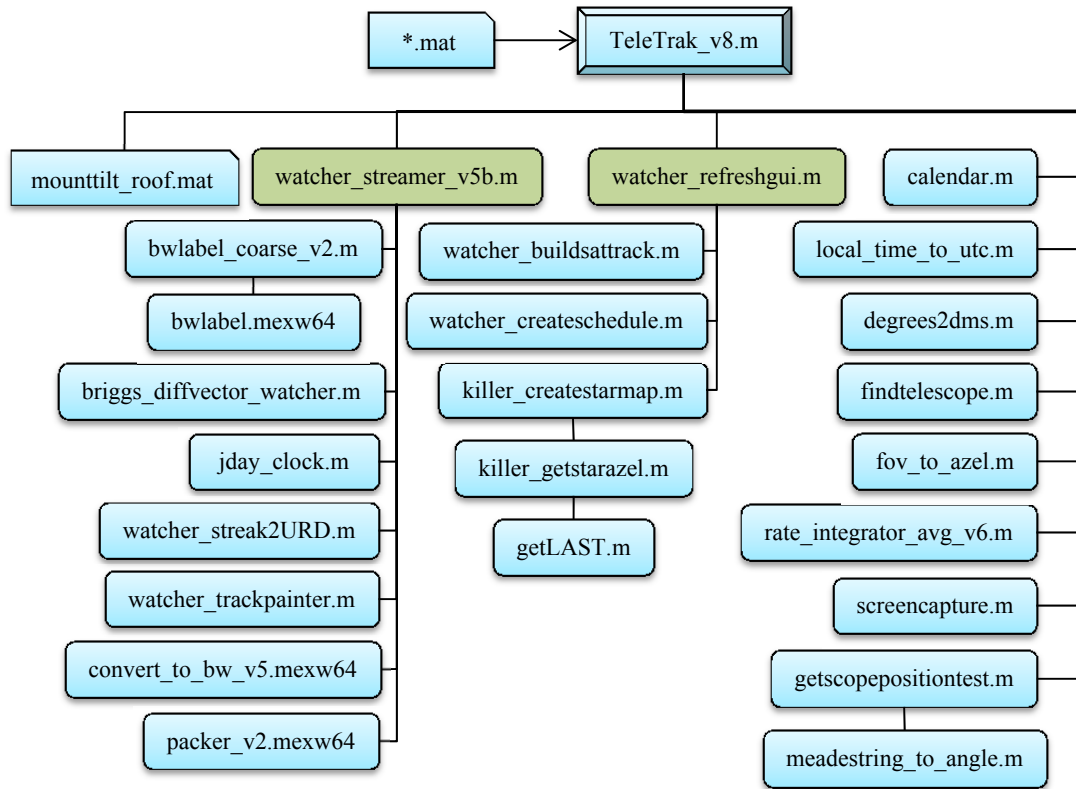


Figure D-3: TeleTrak_v8 Dependency Diagram

Table D-4: TeleTrak_v8 Reference Data File Descriptions

Reference File	Reference Service	Description
mounttilt_roof.mat	TeleTrak_v8.m	Tilt correction matrix file in SEZ frame for AFIT roof telescope

Bibliography

- [1] R. M. Gates and J. R. Clapper, "National Security Space Strategy," January 2011. [Online]. Available: http://www.defense.gov/Portals/1/features/defenseReviews/NSSS/NationalSecuritySpaceStrategyUnclassifiedSummary_Jan2011.pdf. [Accessed 2015].
- [2] C. Calkins, "AFSPC Commander visits 'Team Patrick-Cape'," 12 March 2014. [Online]. Available: <http://www.af.mil/News/ArticleDisplay/tabid/223/Article/473403/shelton-announces-new-space-situational-awareness-satellite-program.aspx>.
- [3] M. Gruss, "Shelton Orders Shutdown of Space Fence," 6 August 2013. [Online]. Available: <http://spacenews.com/36655shelton-orders-shutdown-of-space-fence/>.
- [4] L. Martin, "Construction Underway for U.S. Air Force Space Fence Radar in the Marshall Islands," 23 March 2015. [Online]. Available: <http://www.lockheedmartin.com/us/news/press-releases/2015/march/150323-mst-iwss-construction-underway-for-us-air-force-space-fence-radar.html>.
- [5] M. Whelan, "Space Situational Awareness (SSA) Overview and SSA Sharing," 26 March 2015. [Online]. Available: <https://www.gwu.edu/~spi/USRA%20Brief%20-%20Marty%20Whelan.pdf>. [Accessed 2015].
- [6] "Orbital Debris Quarterly News," July 2015. [Online]. Available: <http://orbitaldebris.jsc.nasa.gov/newsletter/pdfs/ODQNv19i3.pdf>. [Accessed September 2015].
- [7] NORAD, "NORAD Two-Line Element Sets," October 2015. [Online]. Available: <http://www.celestrak.com/NORAD/elements/>. [Accessed October 2015].
- [8] J. L. Gresham, "Optical Closed-Loop Control Tracking With Commercial Telescopes," Air Force Institute of Technology, Masters Thesis, 2012.
- [9] P. Kervin, V. Soo Hoo, D. Nishimoto and D. Liang, "Rapidly deployable Raven-class systems SSA Support in the Field," Air Force Research Laboratory, Space

Vehicles Directorate, Kirtland AFB, NM 87117, 2009.

- [10] B. Visser, C. Sabol and S. Dahlke, "Geosynchronous Orbit Determination Using High Accuracy Angular Observations," *American Astronomical Society (AAS)*, 05-135.
- [11] A. D. Glecker and M. C. Butterfield, "Viral Space Situational Awareness," 2012. [Online]. Available: http://www.amostech.com/TechnicalPapers/2012/Data_Services/GLECKLER.pdf.
- [12] ExoAnalytic, "ExoAnalytic Space Operations Center (ESpOC)," 2014. [Online]. Available: <http://exoanalytic.com/products/espoc>.
- [13] D. Sibert, S. Kelso, and R. Hall, "Collaborative Commercial Space Situational Awareness with ESpOC-Empowered Telescopes," *AMOS tech Conference*.
- [14] "Astronomy Software," AstronomyOnline, 2013. [Online]. Available: <http://astronomyonline.org/astronomysoftware.asp>. [Accessed 2016].
- [15] J. P. Skura and T. H. Choo, "SciBox: A Software Library for Rapid Development of Science Operation Simulation, Planning, and Command Tools," *Johns Hopkins APL Technical Digest*, vol. 25, no. 2, 2004.
- [16] B. Bennett, "Systems Engineering Approach to Automated Cueing For LEO Satellite Tracking," Air Force Institute of Technology, Masters Thesis, 2014.
- [17] M. M. Schmunk, "Initial Determination of Low Earth Orbits Using Commercial Telescopes," Air Force Institute of Technology, Masters Thesis, 2008.
- [18] G. C. Briggs, "Satellite Detection and Real-time Orbit Estimation with Commercial Telescopes," Air Force Institute of Technology, Masters Thesis, 2011.
- [19] D. Moomey, "Aiding Geostationary Space Situational Awareness Using Small Aperture Commercial Telescopes," Air Force Institute of Technology, Masters Thesis, 2015.

- [20] R. Rafeed and C. Paul, "The Linux Development Platform," 2003. [Online]. Available: http://www.faqs.org/docs/ldev/0130091154_25.htm. [Accessed 2016].
- [21] NASA, "Definition of Two-Line Element Set Coordinate System," 2011. [Online]. Available: http://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SOP_Help/tle_def.html. [Accessed 2016].
- [22] Microsoft, "Modularity," 2015. [Online]. Available: [https://msdn.microsoft.com/en-us/library/ff921069\(v=pandp.20\).aspx](https://msdn.microsoft.com/en-us/library/ff921069(v=pandp.20).aspx). [Accessed September 2015].
- [23] "Telescope and Binocular Buying Guide," Celestron, 2015. [Online]. Available: <http://www.wayfair.com/Buying-Guides/Telescope-and-Binocular-Buying-Guide-E3452>. [Accessed 1 October 2015].
- [24] "Orion ShortTube 80 Refractor Telescope Optical Tube Assembly Specifications," 2016. [Online]. Available: http://www.telescope.com/catalog/product.jsp?productId=9948&utm_source=google&utm_medium=comparisonshopping&utm_campaign=US-googlemerchant&gclid=CjwKEAiAoIK1BRcRiMqphvnlw1wSJAAOebPMLJDMKU21cpYca6iRA5mb7v_pdJ3eSC7HjFr5vh7eDhoCCDzw_wcB. [Accessed 2016].
- [25] Woodland Hills Camera & Telescopes, "Astrovid - StellaCam3 Monochrome CCD Video Integrating Imaging System with NTSC Video Output," Astrovid, 2016. [Online]. Available: <http://telescopes.net/store/av-stcam3-stellacam3-monochrome-ccd-video-integrating-imaging-system.html>. [Accessed 2016].
- [26] Sony, "Diagonal 8mm (Type1/2) CCD Image Sensor for EIA B/W Video Cameras," [Online]. Available: <http://www.digchip.com/datasheets/parts/datasheet/453/ICX418ALL.php>.
- [27] "Astrometry.net," National Science Foundation, National Aeronautics and Space Administration, and Canadian National Science and Engineering Research Council, [Online]. Available: <http://astrometry.net/summary.html>. [Accessed 2016].

- [28] Sewell Direct, "Epiphan VGA2USB LR VGA to USB Frame Grabber," Epiphan, 2016. [Online]. Available: <https://sewelldirect.com/epiphan-vga2usb-lr-vga-to-usb-converter>. [Accessed 2016].
- [29] D. T. Kelso, "Orbit Coordinate Systems, Part I," Celestrak, 17 May 2014. [Online]. Available: <https://celestrak.com/columns/v02n01/>. [Accessed 2016].

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 24-03-2016		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From - To) June 2015 - March 2016	
TITLE AND SUBTITLE Development of a modularized software architecture to enhance SSA with COTS telescopes				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) McCafferty, Julian P., 2d Lt, USAF				5d. PROJECT NUMBER 227	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENY-16-M-227	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space Vehicles Directorate, Air Force Research Laboratory Dr. Gregory Spanjers (ST), Chief Scientist 3550 Aberdeen Drive SE Albuquerque, NM 87117				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RV	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT As the catalog of Earth orbiting objects continues to grow exponentially, so too does the necessity for Space Situational Awareness (SSA). Previous work at AFIT has explored augmenting the Space Surveillance Network (SSN) by demonstrating detection and tracking of orbiting objects using Commercial-Off-The-Shelf (COTS) telescopes and Air Force generated Two-Line Element Sets (TLE). This research explores the process of developing and reengineering code into a modularized, hierarchical component architecture designed for the end user while enabling developers to continue to modify the software for future applications. Three graphical user interfaces (GUI) are compiled into standalone executable programs using MATLAB for propagating orbits, mapping, and performing observations. Previous AFIT in-house MATLAB code is further developed to be hardware agnostic and continue to operate on future operating systems. Finally, open-loop optical tracking of Low Earth Orbiting (LEO) satellites is demonstrated using a Meade LX200GPS telescope with 100% of targets captured in the Orion 80mm spotting scope. The ultimate result of this work is a set of software tools available to students and researchers which can be further developed to realize a cost saving technology to meet the Air Force's growing demand for SSA.					
15. SUBJECT TERMS Space Situational Awareness, Modularized, Software Architecture, Telescope, Tracking, COTS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 114	19a. NAME OF RESPONSIBLE PERSON Dr. Richard Cobb, AFIT/ENY
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937) 255-3636, ext 4559 (Richard.Cobb@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18