



**AUTONOMOUS ASSOCIATION OF GEO RSO OBSERVATIONS USING DEEP  
NEURAL NETWORKS**

THESIS

Ian W. McQuaid, First Lieutenant, USAF

AFIT-ENG-MS-18-S-014

**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, the 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-ENG-MS-18-S-014

AUTONOMOUS ASSOCIATION OF GEO RSO OBSERVATIONS USING DEEP  
NEURAL NETWORKS

THESIS

Presented to the Faculty  
Department of Electrical and Computer Engineering  
Graduate School of Engineering and Management  
Air Force Institute of Technology  
Air University  
Air Education and Training Command  
in Partial Fulfillment of the Requirements for the  
Degree of Master of Science in Computer Science

Ian W. McQuaid, BS  
First Lieutenant, USAF

September 2018

DISTRIBUTION STATEMENT A:  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AFIT-ENG-MS-18-S-014

AUTONOMOUS ASSOCIATION OF GEO RSO OBSERVATIONS USING DEEP  
NEURAL NETWORKS

Ian W. McQuaid, BS  
First Lieutenant, USAF

Approved:

---

Laurence D. Merkle, PhD (Chairman)

---

Date

---

Brett J. Borghetti, PhD (Member)

---

Date

---

Richard G. Cobb, PhD (Member)

---

Date

---

Kenneth M. Hopkinson, PhD (Member)

---

Date

**Abstract**

Ground-based non-resolved optical observations of resident space objects (RSOs) in geosynchronous orbit (GEO) represent the majority of the space surveillance network's (SSN's) deep-space tracking. Reliable and accurate tracking necessitates temporal separation of the observations. This requires that subsequent observations be associated with prior observations of a given RSO before they can be used to create or refine that RSO's ephemeris. The use of astrometric data (e.g. topocentric angular position) alone for this association task is complicated by RSO maneuvers between observations, and by RSOs operating in close proximity. Accurately associating an observation with an RSO thus motivates the use of photometric light curves in that association process. Contemporary machine learning, specifically deep neural networks (DNNs), offers mechanisms to perform that association autonomously by first learning patterns between observations and their parent RSOs. This research assesses the extent to which a trained DNN can autonomously associate previously unseen observations with the RSO they represent. DNN performance is assessed by recording the percentage of observations from a held-out testing set associated to the correct RSO. The performance of the DNN is evaluated in several simulated scenarios. In the first, the RSOs maintain station for one year. In the second, the RSOs station keep in progressively closer proximity. In the third, the RSOs permute stations with one another. In each case the photometric observations are generated as if observed by the Maui GEODSS site. This research contributes to the development of autonomous SSN telescopes, systems to distinguish closely spaced RSOs, and systems to autonomously and rapidly update an RSO's ephemeris after it maneuvers.

## Table of Contents

|                                                             | Page  |
|-------------------------------------------------------------|-------|
| Abstract . . . . .                                          | iv    |
| Table of Contents . . . . .                                 | v     |
| List of Figures . . . . .                                   | viii  |
| List of Tables . . . . .                                    | x     |
| <br>I. Introduction . . . . .                               | <br>1 |
| 1.1 Motivation . . . . .                                    | 1     |
| 1.2 Background . . . . .                                    | 3     |
| 1.3 Research Objective . . . . .                            | 4     |
| 1.3.1 Research Questions . . . . .                          | 4     |
| 1.4 Approach . . . . .                                      | 5     |
| 1.5 Assumptions and Limitations . . . . .                   | 7     |
| 1.6 Thesis Overview . . . . .                               | 8     |
| <br>II. Background and Related Work . . . . .               | <br>9 |
| 2.1 Overview . . . . .                                      | 9     |
| 2.2 Space Situational Awareness . . . . .                   | 9     |
| 2.2.1 Optics and Overview . . . . .                         | 10    |
| 2.2.1.1 Earth Centered Inertial Coordinate System . . . . . | 10    |
| 2.2.1.2 Visual Magnitude . . . . .                          | 11    |
| 2.2.1.3 Space Surveillance Network . . . . .                | 12    |
| 2.2.2 GEO Track Creation . . . . .                          | 15    |
| 2.2.3 Orbit Determination . . . . .                         | 16    |
| 2.2.3.1 Laplace’s Method . . . . .                          | 17    |
| 2.2.3.2 Gauss’ Method . . . . .                             | 18    |
| 2.2.3.3 Partial Knowledge Methods . . . . .                 | 19    |
| 2.2.3.4 Admissible Regions . . . . .                        | 19    |
| 2.2.4 Observation and Track Association . . . . .           | 20    |
| 2.2.5 Sensor Scheduling . . . . .                           | 22    |
| 2.2.6 Photometry . . . . .                                  | 24    |
| 2.2.7 Satellite Maneuvers . . . . .                         | 25    |
| 2.2.7.1 Station Keeping Maneuvers . . . . .                 | 25    |
| 2.2.7.2 Rendezvous and Proximity Maneuvers . . . . .        | 27    |

|                                                              | Page |
|--------------------------------------------------------------|------|
| 2.3 Machine Learning . . . . .                               | 29   |
| 2.3.1 Overview and Definitions . . . . .                     | 30   |
| 2.3.2 Classification . . . . .                               | 31   |
| 2.3.3 Artificial Neural Networks . . . . .                   | 32   |
| 2.3.4 Convolutional Neural Networks . . . . .                | 34   |
| 2.3.4.1 Overview . . . . .                                   | 34   |
| 2.3.4.2 Convolution and Pooling . . . . .                    | 34   |
| 2.3.4.3 Deep Convolutional Neural Networks . . . . .         | 36   |
| 2.3.5 Time Series Classification . . . . .                   | 36   |
| 2.4 Summary . . . . .                                        | 37   |
| III. Approach and Methodology . . . . .                      | 39   |
| 3.1 Overview . . . . .                                       | 39   |
| 3.2 Dataset Generation . . . . .                             | 41   |
| 3.3 ANN Topology . . . . .                                   | 46   |
| 3.4 HPC Resources . . . . .                                  | 48   |
| 3.5 Baseline Association of RSOs . . . . .                   | 49   |
| 3.6 Association of Closely Spaced RSOs . . . . .             | 51   |
| 3.7 Association of Maneuvering RSOs . . . . .                | 53   |
| IV. Results . . . . .                                        | 56   |
| 4.1 Baseline Association of RSOs . . . . .                   | 56   |
| 4.2 Association of Closely Spaced RSOs . . . . .             | 59   |
| 4.3 Association of Maneuvering RSOs . . . . .                | 60   |
| V. Conclusions and Recommendations for Future Work . . . . . | 63   |
| 5.1 Conclusions . . . . .                                    | 63   |
| 5.2 Recommendations . . . . .                                | 64   |
| Appendix A: Load Ephemeris . . . . .                         | 66   |
| Appendix B: Generate Data . . . . .                          | 69   |
| Appendix C: Keras Experiment . . . . .                       | 74   |
| Appendix D: Cluster Experiment . . . . .                     | 78   |
| Appendix E: Process Results . . . . .                        | 82   |

|                                                              | Page |
|--------------------------------------------------------------|------|
| Appendix F: Baseline Experiment Code . . . . .               | 84   |
| Appendix G: Closely Spaced Objects Experiment Code . . . . . | 90   |
| Appendix H: Maneuvering Experiment Code . . . . .            | 96   |
| Bibliography . . . . .                                       | 102  |

## List of Figures

| Figure                                                                                                                                                                                                                                                                                                                                                                     | Page |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 2.1 Diagrammatic description of ECI coordinates. Right ascension is measured moving east from the vernal equinox, and declination measured either north or south from the celestial equator. Both are fixed to the background stars instead of the Earth [54]. . . . .                                                                                                     | 11   |
| 2.2 Graphical depiction of a Hohmann transfer orbit [33]. . . . .                                                                                                                                                                                                                                                                                                          | 28   |
| 2.3 Artificial neural network with one hidden layer [15]. . . . .                                                                                                                                                                                                                                                                                                          | 33   |
| 2.4 Topology of a typical convolutional neural network. Large squares correspond to intermediate representations. Small squares correspond to the following layer's kernel to be convolved with the current layer. Squares are stacked to indicate the individual filters learned (sometimes called depth). Subsampling is synonymous with pooling [8]. . . . .            | 35   |
| 3.1 Diagram of process used in this research. From left to right ephemeris are created for the experiment in question, then photometric aspects are modeled and observations produced. Afterwards various cluster resources are used to evaluate a collection of ANNs on those observations, and the results of the experiment are produced using those resources. . . . . | 40   |
| 3.2 Graphical depiction of the "Nadir alignment with orbit normal constraint" attitude profile. From AGI's website at: <a href="http://help.agi.com/stk/index.htm">http://help.agi.com/stk/index.htm</a> . . .                                                                                                                                                             | 42   |
| 3.3 Graphical depiction of the "Nadir alignment with sun constraint" attitude profile. From AGI's website at: <a href="http://help.agi.com/stk/index.htm">http://help.agi.com/stk/index.htm</a> . . . . .                                                                                                                                                                  | 42   |
| 3.4 Example plot of light curve over a 6 second observation. Signal is sampled at 1000Hz. . . . .                                                                                                                                                                                                                                                                          | 44   |

| Figure                                                                                                                                                                                                                                     | Page |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 3.5 Diagram of convolutional neural network topology used to associate photometric observations to RSO . . . . .                                                                                                                           | 48   |
| 3.6 Station keeping catalog used in baseline association experiment. . . . .                                                                                                                                                               | 50   |
| 3.7 Station keeping catalog used in closely spaced objects experiment. . . . .                                                                                                                                                             | 52   |
| 3.8 Maneuvering catalog where RSOs permute stations every 31 days. Used in the maneuvering experiment. . . . .                                                                                                                             | 54   |
| 4.1 Mean accuracy arranged by manner in which each RSO pair differs. Bar height gives mean accuracy given those differences, and error bars show standard deviation. Note that each bar describes 4 RSO pairs, hence a sample of size 120. | 58   |
| 4.2 Mean accuracy by RSO pairing differences. Bar height gives mean accuracy given those differences, and error bars show standard deviation. Note that each bar describes 4 RSO pairs, hence a sample of size 120. . . . .                | 61   |

## List of Tables

| Table                                                                                                                                                                          | Page |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 2.1 Sample maneuver schedule to minimize individual maneuver magnitudes [13].                                                                                                  | 27   |
| 3.1 Table of values used in STK model of GEODSS space object identification<br>CCD [5]. . . . .                                                                                | 43   |
| 3.2 Hokulea compute cluster characteristics and performance [2]. . . . .                                                                                                       | 49   |
| 3.3 RSO shapes, materials, and attitude profiles used in this experiment. . . . .                                                                                              | 51   |
| 3.4 Longitude of assigned station for each RSO in the closely spaced RSOs scenario.                                                                                            | 52   |
| 4.1 Accuracy means and standard deviations for every RSO pairing in the baseline<br>association experiment. Each cell summarizes a sample of size 30. . . . .                  | 57   |
| 4.2 Accuracy means and standard deviations for every RSO pairing in the closely<br>spaced objects association experiment. Each cell summarizes a sample of size<br>30. . . . . | 59   |
| 4.3 Accuracy means and standard deviations for every RSO pairing in the<br>maneuvering association experiment. Each cell summarizes a sample of size 30.                       | 60   |

# AUTONOMOUS ASSOCIATION OF GEO RSO OBSERVATIONS USING DEEP NEURAL NETWORKS

## I. Introduction

### 1.1 Motivation

Space situational awareness (SSA) is a key Air Force mission that, as the name implies, requires active awareness of all objects in the Earth's orbit. This is important to national security, as drifting objects could cause collisions, or adversaries could maneuver objects so as to capture potentially confidential or classified data. The set of resident space objects (RSOs) in geosynchronous Earth orbit (GEO) are of particular relevance, as the GEO belt both offers an ideal location for satellites to observe the Earth, and the unoccupied area within the GEO belt is becoming more and more scarce as additional GEO satellites are launched. Furthermore, observing GEO RSOs is particularly challenging, principally because of how far it is from the Earth. Reaction to events necessarily takes much longer, as transit to GEO takes longer than to lower altitude regimes such as LEO. Most often GEO RSO observations are from ground-based electro-optical sensors and consist of non-resolved images. To further complicate this problem, those sensors can only observe RSOs while the sensor is in darkness and sky conditions are clear. When those conditions are met, the sensor can reliably observe GEO RSOs. That being said, tracking them adds another level of complexity.

Tracking an RSO requires many observations. More complicated still, one requires observations separated in time so as to better approximate the orbit. In an operational context this is further complicated by sensor scheduling concerns. Realistically, RSOs must be observed hours later or even on a subsequent night. This sadly creates an entirely

new problem: how does one know that they are observing the same RSO? Association is sometimes called correlation, and observations and tracks without a known “parent” RSO are sometimes called uncorrelated observations and tracks (UCOs and UCTs). Association is where one labels an observation with either the track to which it belongs, or a pool of observations to which it belongs. Technically associating to a true RSO identity is termed identification, though here the distinction is blurred as labeling an observation with the correct identity is one manner of associating those observations together.

Current association techniques can be quite effective, especially in the typical day-to-day catalog maintenance use-case. The literature contains a myriad of effective ways to accomplish this [50], [51], [26], [16]. The main limitation of these techniques is that they are all astrometric based. That is, they make their decisions based on the observed kinematics of an RSO, which for non-resolved GEO imagery means angular position in some topocentric coordinate system. Associating observations via angles has two main drawback. First, as RSOs move close together it becomes more difficult to differentiate those observations. At a certain point the two would look, from an astrometric point of view, as if they were observations from the same orbit. The second case is, in a manner, the extreme of the first. Suppose between observations that two RSOs swapped positions. Astrometrically there is no way to detect that, and certainly no way to determine which RSO is which. This scenario may seem contrived, however if one considers performing the SSA mission in the presence of adversarial RSOs, then it would seem reasonable for an RSO to aim to deceive current association systems. In that case, the importance of being robust to such deception is apparent.

A potential solution is to develop an association capability that is not simply astrometric based. Here this research proposes a photometric-based technique. Photometry pertains to, as the name implies, observation and quantification of the visible light reflected to a sensor by an RSO [9]. The following sections describe the goals of this research, the

questions and hypothesis under consideration, as well as a brief overview of the remainder of this document.

## **1.2 Background**

After Operation Desert Storm, the US military came to appreciate the value of space assets in accurate target identification, better damage assessment, and improved communication. Because of this, an increasing portion of the national defense budget has been allocated to refining these capabilities. Adversaries, then, naturally consider space assets themselves, as well as counter-space assets. SSA arises as a necessary consequence of the contested nature of space operations. McCall defines SSA as “the enabling of a description of the location and operation of US space assets as well as the location and function of the assets of other nations, particularly those that are, or could become, our enemies” [36]. This most obviously means that the US must be aware of the locations of all objects orbiting the Earth. It also means that the US must have an understanding of the function of each of those objects [36].

GEO is of particular value for communications, navigation, and surveillance due to its unique quality of being synchronized with the rotation of the Earth. An increasingly populated GEO highlights the importance of SSA in this orbit regime. To that end the Air Force continues to invest in sensors across the globe with the intent of surveying the entirety of GEO. Furthermore, these sensors are designed to detect ever smaller objects so that even small pieces of debris, or otherwise inconspicuous surveillance satellites, are continually tracked [36].

Additionally, orbital debris is an ever deepening concern. The fear is that collisions could cascade and give rise to what is known as Kessler’s Syndrome: a state in which safe space operations becomes impossible due to the vast quantity of debris in LEO. Such a state is not desirable by the US, nor the majority of the world. Even small scale collisions in GEO would result in unusable stations, causing communication and navigation issues

over the effected regions. All of this to say that though SSA is first a national defense concern to the DoD, it also has an impact on world economy and lifestyle [36], [29].

### **1.3 Research Objective**

The objective of this study is to demonstrate the feasibility of association GEO RSO observations using deep neural networks. Additionally, this study will explore cases where that association is more difficult, such as with RSOs operating in close proximity or that maneuver. Though the main intent is as a proof of concept for photometric based association techniques, a rudimentary understanding of the limits of this approach is valuable. If successful this would serve as a technique to leverage when astrometric-only association fails. To that end, this research hypothesizes that deep neural networks (DNNs) trained on photometry can be applied to GEO SSA to address current UCO and UCT association ambiguities. The following discussion is guided by three research questions:

- Are there any cases where a DNN can use photometry to correctly associate observations to the corresponding RSO?
- Can a DNN use photometry to associate observations of RSOs even if those RSOs are closely spaced?
- Can a DNN use photometry to associate observations of RSOs even if those RSOs are maneuvering?

These questions are briefly discussed in turn.

#### ***1.3.1 Research Questions.***

The first research question asks if there are cases where a DNN can use photometry to correctly associate observations to the corresponding RSO. This is obviously quite loose, as certainly some scenarios will exist where such a technique will be effective. That being said the spirit of this question lies in discovering what those cases are. An understanding of what kinds of RSOs one could reasonably expect to differentiate between implies to

what extent to which this could be used in practice. It also bounds the extent to which the more extreme cases considered here could be true. Provided two RSOs fall into the class of distinguishable RSOs constructed by the answer to this question, is it then possible for that technique to hold in more dynamic scenarios?

The second research question asks if a DNN can use photometry to associate observations of RSOs even if those RSOs are closely spaced. The answer depends on the impact of RSO proximity on the differences between two RSO's photometric signature. Provided those differences are not impacted significantly by their proximity, then it seems reasonable to expect them to be distinguishable. More importantly, such a technique would represent additional capability provided by the success of this research. As associating UCOs and UCTs of closely spaced RSOs is a known limitation of existing observation association techniques, photometric-based association would be a valuable addition to the community's analysis tools.

The third research question asks if a DNN use photometry to associate observations of RSOs even if those RSOs are maneuvering. In a traditional tracking or maintenance scenario this situation may not be reasonable, however this question is intended to address the aforementioned concerns of adversarial RSO behavior. In that case, astrometric based techniques are simply not viable. The implications of accurate SSA in the presence of adversarial deception are profound. Unfortunately, the complexity involved in such a system would be just as profound, so the previous questions serve to scope and quantify this research. With those limits in mind, this research intends to determine to what extent UCOs/UCTs association of maneuvering RSOs is possible.

## **1.4 Approach**

In order to address the aforementioned research questions, appropriate datasets must be obtained, a deep neural network designed, and then that network must be trained and evaluated using that dataset. A separate dataset is generated for each research question.

In this study the data will be simulated using AGI's Systems Tool Kit (STK). The deep neural network will be designed using the Keras Python front-end, and then training and evaluation will take place on the Maui High Performance Computing Center's (MHPCC's) system [12], [2].

The data is simulated via STK so as to model both a realistic RSO ephemeris, as well as to model the optics necessary to produce realistic photometric light curves at each observation time. For each hypothesis a year-long STK scenario is constructed and populated with a representative collections of satellites. The Astrogator package is used to model the manner that those satellite's maneuver throughout the year, be that station keeping or otherwise. The EOIR package is used to model the telescope and CCD at the Maui Ground-based Electro-Optical Deep Space Surveillance (GEODSS) site as well as each RSO's photometric properties. EOIR is then used to produce the simulated photometric observations at specified times throughout the year. Each scenario contains 8 satellites, all of which are observed approximately 850 times throughout each scenario. Each individual observation is a 6 second long 1000 Hz signal of observed object intensity over the period of the observation.

Keras is then used to create a DNN whose input is a 1-dimensional single-channel signal of length 6000 [12]. This network is designed for only two class problems. That is, it will always be distinguishing some two RSOs from the scenario. Experimentally, all possible pairings will be considered, though a new DNN will be created and trained for every pair of RSOs to be distinguished. By noting which networks are effective and which are ineffective, one can learn what qualities make two RSOs distinguishable. MHPCC's high performance computing resources are used so that an appropriate amount of cross-validation, as well as experiment replications, can be performed so as to give results statistical significance. Those results are finally aggregated and discussed for each scenario.

## 1.5 Assumptions and Limitations

Assume we can detect everything we look at

Assume away sensor scheduling, but add a minimum time between observations

Assume everything is GEODSS

It should first be noted that the difference between track association and identification is non-trivial. In all cases considered here the object identities are known a priori, so all associations are performed against known objects. Hence identification is a more appropriate term. With that in mind, the distinction is disregarded here as classification by an hypothetical RSO label can be thought of as association. Additionally this research identifies observations as opposed to tracks. The key difference is that initial orbit determination is not performed at any point, so tracks are never constructed. Instead this research considers the problem of labeling the observations themselves with their corresponding RSO.

The scenarios used in this research are all 1 year long starting 1 Jan 2019 at 1200 UTC and lasting until 1 Jan 2020 at 1200 UTC. This choice was so that at any point throughout the course of this research a more current ephemeris could be used instead and would still be defined on the entire scenario period. The 1 year length was chosen so that RSO patterns of life would be observable.

Atmospheric effects are not accounted for, due to stability issues with STK when running EOIR in conjunction with the Modtran atmosphere model.

The paradigm of daily burns is not necessarily employed in many or all operational systems, though is used in this research for reproducibility. Any schedule that amounts to an RSO maintaining appropriate GSO station will correspond to a representative ephemeris. The main advantage of the daily burn schedule over another is that daily burns will cause less disturbance to on-board equipment, an advantage that is not relevant to this research.

RSOs not within angular resolution of sensor.

## **1.6 Thesis Overview**

The remainder of this document is aimed at addressing the research hypothesis and questions presented here. First, Chapter 2 provides necessary background in both SSA as well as machine learning. Chapter 3 contains details of the methodology and experiment design used to address these questions. Chapter 4 documents the results as well as discussion and immediate findings from those results. Finally, Chapter 5 presents the conclusions reached and recommendations for future research.

## **II. Background and Related Work**

### **2.1 Overview**

Before this research’s methodology is described a certain amount of background is necessary. This chapter covers the basics of both SSA and machine learning. The former gives context to the simulation used to generate data, as well as the specific practical application being addressed in this research. The latter gives insight into the capability of state-of-the-art machine learning algorithms to model complex relationships. An understanding of both domains will facilitate understanding of this research’s hypothesis and thus motivate the methodology used to address that hypothesis.

### **2.2 Space Situational Awareness**

In order to understand this research’s goals it is first necessary to understand the SSA domain. SSA is comprised of many individual requirements, each with their own metrics to optimize, and often these are at odds with one another. Coverage (the proportion of space near the Earth that is actively monitored), detection (the discovery of RSOs), tracking (the orbit estimation of RSOs), and characterization (understanding the function of RSOs) are several such metrics. Here we mostly disregard the coverage problem, but cover detection and tracking, which are central to this research. First, basic coordinate systems used, optics terminology, and a catalog of existing sensors are discussed. Afterwards a basic description of the GEO track creation process is given to provide context to the task this research aims to automate. Next the theory and techniques behind orbit determination of non-resolved optical tracks are given. In addition to determining orbit parameters, photometric RSO data and its potential applications are discussed. Finally this section concludes by discussing satellite maneuvers for both station keeping as well as rendezvous and proximity operations.

### ***2.2.1 Optics and Overview.***

#### ***2.2.1.1 Earth Centered Inertial Coordinate System.***

A standard system for expressing RSO location is vital for any meaningful discussion of SSA. Most commonly, a coordinate system not fixed to the Earth's rotation is used to remove complexities associated with that rotation. The coordinate system is instead fixed to the background stars with the origin at the center of the Earth. The celestial sphere is the geometric reference frame typically used in astronomy, and is essentially a non-rotating Earth of infinite radius with the poles and equator fixed to the background stars. The stars in our night sky can be thought of as lying somewhere on the surface of this sphere, giving astronomers a geometric construct to describe positions. The celestial equator and poles are the same as those of the Earth at either of the year's equinox; the specific times in the year where the Sun is positioned over the equator.

The Earth Centered Inertial (ECI) coordinate system describes object positions on the celestial sphere with two angles: termed right ascension ( $\alpha$ ) and declination ( $\delta$ ). Figure 2.1 shows the ECI coordinate system graphically. Right ascension is measured to the east along the celestial equator starting at the vernal equinox (also called the first point of Ares). Usually this is expressed from  $0^\circ - 360^\circ$ , though at times can be given in hr-m-s. The declination is measured from the celestial equator and given from  $-90^\circ - 90^\circ$ ; objects to the north of the celestial equator are given positive declinations and objects to the south negative declinations [37]. This is in contrast to traditional spherical coordinates in two ways. First that no range information is provided as it is often not known or meaningful in astronomic applications, and second in that the vertical angle is given with respect to the equator rather than the vertical axis. ECI is a common coordinate system used by ground-based electro-optical telescopes. Though often the phrases "astrometric data" or "astrometric observation" are used throughout this document, ECI is often the form that data takes.

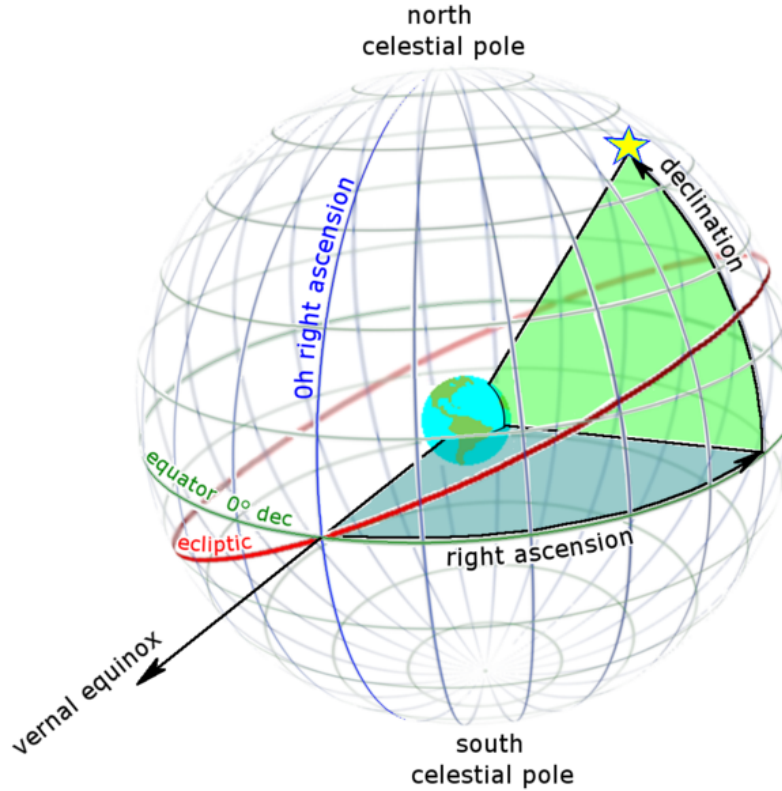


Figure 2.1: Diagrammatic description of ECI coordinates. Right ascension is measured moving east from the vernal equinox, and declination measured either north or south from the celestial equator. Both are fixed to the background stars instead of the Earth [54].

### 2.2.1.2 Visual Magnitude.

Visual magnitude, also called apparent magnitude, is a measure of object brightness. It originates with the Hellenistic practice of categorizing stars into six magnitudes, with stars of the first magnitude being the brightest [45]. This was, of course, subjective at the time, as no quantifiable measures of brightness existed. When the system was later formalized in the 19th century, it was constructed as a logarithmic scale with first magnitude stars being 100 times brighter than sixth magnitude stars [45]. Polaris was originally used as the reference brightness, however scientists later discovered that Polaris's brightness varied

significantly over time. Instead Vega is used, and corresponds to visual magnitude 0 for visible and near-infrared wavelengths in the modern system [45].

This is an inverse scale; larger magnitudes correspond to dimmer objects. For reference, the sun, as seen from Earth, is about magnitude  $-26.74$  and the full moon about magnitude  $-12.90$ . The average human eye is capable of seeing objects as dim as magnitude 6.5, and as bright as magnitude  $-25.0$  without experiencing pain. GEO surveillance optical sensors routinely search and find objects as dim as magnitude 17 [45]. This is a dominant concern in RSO detection. In this research it is assumed that each RSO can be detected, so this information serves as context to the RSO tracking process.

### ***2.2.1.3 Space Surveillance Network.***

A source of reliable and accurate observations of space objects is critical to the SSA mission. Several existing ground-based and space-based systems enable object detection, tracking, and identification. These elements of the Space Surveillance Network (SSN) include:

- Ground-based Electro-Optical Deep Space Surveillance (GEODSS)
- Space Surveillance Telescope (SST)
- Space Based Space Surveillance (SBSS)
- Sapphire
- ORS-5

These SSN elements are discussed in turn.

Ground-based Electro-Optical Deep Space Surveillance (GEODSS) sensors are the primary source for US ground-based SSA. GEODSS consists of three active sites at Diego Garcia, Maui, and Socorro. Each of these sites hosts three optical telescopes, each with a 1 m aperture, a 2.15 m focal length, and a 2 square degree instantaneous field of view. The image is formed via charge coupled device (CCD) with  $1960 \times 2560$  pixels and a  $24 \mu\text{m}$

pitch (i.e. pixel separation). GEODSS is effective and highly accurate, but does not cover the entire GEO belt. Notably a wide band over western Europe and a narrow band over eastern Asia are not observable via GEODSS. [27]

A newer US ground-based SSA asset is the Space Surveillance Telescope (SST). The Air Force took command of this system in October 2016, after it was developed by DARPA and installed at the White Sands Missile Range in New Mexico. It is scheduled to be moved to Harold E. Holt Naval Communication Station in Western Australia where it will be owned by the United States Air Force, but operated and maintained by the Australian government. This is perhaps the most impressive ground-based telescope in the SSN constructed to date, with a 3.5 m aperture and  $f/1.0$  (meaning the focal length is also 3.5 m) attached to a  $2k \times 4k$  CCD. This enables detection of objects as dim as magnitude 18. [34], [3]

Space Based Space Surveillance (SBSS) is, as the name implies, a space-based SSA asset. The platform itself is in a sun-synchronous orbit at approximately 630 km altitude. A sun-synchronous orbit is a near-polar orbit where the satellite passes over a given location on the Earth at approximately the same local mean solar time each day. This results in the satellite being in near constant sunlight. SBSS leverages this by placing the sensor on a two-axis articulated mount so that the sensor has a  $3\pi$  steradians field of view, all the while leaving the satellite body and solar panels oriented towards the Sun. The sensor itself has a 30 cm aperture. Ackermann supposes that the CCD is similar to one used on the Kepler mission (both manufactured by Ball Aerospace). Under that assumption, the sensor would have a  $2200 \times 1044$  pixel CCD on a  $27 \mu\text{m}$  pitch, and a 2 degree by 4 degree instantaneous field of view. Though data is not publicly available, aperture scaling implies that SBSS should be able to detect objects as dim as 16.5 visual magnitude. SBSS has two modes: rate-track and sidereal-track. The former tracks a known satellite's location causing background stars to appear as streaks and allowing it to see dimmer objects at the

expense of astrometric accuracy. The latter tracks background stars, causing the object to appear as a streak. In this mode the sensor won't be as sensitive, but will produce more accurate astrometric estimations. [34], [4]

Sapphire is a Canadian satellite that contributes to the US SSN. In many ways it is like SBSS. It is in a sun-synchronous orbit at a higher altitude than SBSS (786 km) and capable of both rate and sidereal track. The sensor itself is smaller, at only 15 cm aperture with a  $1.4 \times 1.4$  degree instantaneous field of view. The dimmest objects that Sapphire can see are 15 visual magnitude. The sensor also requires reorientation of the entire satellite body, rather than gimbaling the sensor separately as with SBSS. This means that SBSS can produce more observations over a span of time than Sapphire. That being said, Sapphire is considerably less expensive at \$96 M compared to SBSS at \$850 M, so the performance difference is considered by many to be an acceptable trade-off. [34] [46]

ORS-5 is a LEO satellite launched August 2017. Conceptually, the satellite leverages the novel Geometry Optimized Space Telescope (GeOST) concept. In essence, it stares from LEO to a portion of the GEO belt ahead of itself such that the object and itself are laterally stationary relative to one another. Because of this, the observed objects stay on the same CCD pixels for longer, amounting to longer integration times so that dimmer objects can be seen without the need for a larger aperture. Analysts claim an order of magnitude sensitivity (i.e. an additional unit of visual magnitude) improvement with this technique, giving ORS-5 sensitivity hypothetically equivalent to SBSS with only a 10 cm aperture, and at a cost comparable to Sapphire ( $\sim$  \$80 M). The most notable drawback of this system is that it can only observe a small portion of the GEO belt at any given time. [34]

Regarding this research, it suffices to say that GEO RSOs are observed by a myriad of sensors with widely varying capabilities. GEODSS is used here simply for standardization purposes and, though one hopes the conclusions drawn by this research are agnostic of

sensor, it should be kept in mind that results could differ significantly if the observation source changes.

### ***2.2.2 GEO Track Creation.***

Notionally, a track (also called an ephemeris) is an estimation of an RSOs position and velocity vectors. Before a track can be created, one would first need to detect a GEO RSO. Generally speaking, this is via optical sensors. These sensors will report the light intensity they capture over each pixel in their field of view, though for GEO objects this can rarely be resolved to an image. One might then attempt to detect objects via radar instead of optical telescopes, however radar systems that have frequency and power sufficient to detect a GEO RSO also have angle uncertainties sufficiently large as to limit the utility of those detections. So it would seem that the task of detecting and tracking GEO RSOs must be relegated to the non-resolved imagery of electro optical telescopes [34].

The manner in which this is done differs by sensor, though here GEODSS will be used as an exemplar [59]. A detection or observation (not to be confused with the machine learning term of the same name) is a single light streak usually several seconds long and consisting of several frames of data. Each frame consists of the pixel (i.e. CCD) outputs, where the units are usually count of electrons contained in each pixel. Intuitively, longer frames correspond to detection of dimmer objects. The dimmest object that a sensor can detect is referred to as the sensitivity of the sensor. Unfortunately, longer frames also give the object more time to move around within a given pixel, as well as between pixels, resulting in more astrometric uncertainty. This can be mitigated by having the sensor rate-track.

As far as the observations themselves, one could obtain angular positions by recognizing the star field behind and around the object in the non-resolved imagery produced by the sensor. This yields an astrometric estimate of the object; that is, a prediction of the RSO's position and velocity. Though technically astrometry means any

kinematic characterization of the object, it typically means angles-only predictions of object location based on the position of the light streak relative to the background stars. A single observation, in GEODSS case, means an astrometric estimation at the beginning and end of the aforementioned light streak.

A set of observations can be constructed into an object track via either performing an initial orbit determination from the obtained angle estimates, or by improving upon an existing ephemeris using the newly obtained angle measurements. In general, the principle is that greater angular separation in observations corresponds to more accurate ephemerides. Of course one must then be sure that the observations were created by the same object. Thus non-resolved observations must first be associated to one another before a track can be created from them. Current track association techniques are based primarily on astrometry and thus are limited in two major ways. First, even in a static catalog, GEO RSOs spaced close together or with orbits that pass close together could be erroneously associated with each other. Second, if objects choose to maneuver, it is not clear how one would associate future observations of that object. A more detailed discussion of current association techniques is given later in this chapter.

### ***2.2.3 Orbit Determination.***

Generally speaking, orbit determination can either refer to the process of producing an initial approximation of the object's orbital parameters, or to the process of refining those parameters with additional observations and some form of optimization algorithm. Here this research assumes the former, as the assumption is that we are unaware of the RSOs identity, and thus have no initial orbital parameters. Discovery of a new space object is typically followed by characterizing the object and performing initial orbit determination. Historically, astronomers have focused on celestial objects, such as planets, moons, or asteroids, but in recent years the focus has shifted to objects orbiting the Earth. As these techniques were exercised in practical applications, it becomes evident that three

observations were not sufficient for accurate initial orbit determination. Accurate initial orbit determination techniques require many observations over a wide angular spread of the RSO's orbit. Several initial orbit determination methods are used by astronomers, including Laplace's method and Gauss' method. These both have their strengths and weaknesses, which are addressed in the following sections. [53]

### 2.2.3.1 *Laplace's Method.*

In 1780 Laplace devised the first general purpose initial orbit determination technique. Laplace developed the algorithm directly from the equations of motion. Starting with the geocentric (Earth's center at the origin) location of the object and the observer as  $\mathbf{r}$  and  $\rho$  respectively, define the topocentric (observer at the origin) location of the object as  $\mathbf{R} = \mathbf{r} - \rho$ . Then from the equations of motion for a massless satellite

$$\ddot{\mathbf{r}} = -\frac{GM_E \mathbf{r}}{r^3} + \mathbf{P}$$

where  $\mathbf{P}$  represents the disturbing force per unit mass. Now we introduce an observed topocentric position vector  $\mathbf{l}$  to this equation. Here  $\mathbf{l}$  is a unit vector representing our angular measurements of an object's position, whereas  $\mathbf{R}$  represents both angle and range to the object. Substitution of  $\mathbf{l}$  into the previous equation gives

$$\ddot{\mathbf{R}}\mathbf{l} + 2\dot{\mathbf{R}}\dot{\mathbf{l}} + \mathbf{R}\ddot{\mathbf{l}} = -\frac{GM_E \mathbf{r}}{r^3} + \mathbf{P}$$

This differential equation can be solved for  $\mathbf{R}$  provided values of  $\mathbf{l}$  for at least 3 different times. This would yield a polynomial in  $\mathbf{R}$  of degree 8, from which one could derive the remainder of the orbital parameter set.

This will indeed produce estimates of an object's orbit, but the process is prone to imprecision. In particular, numeric differentiation is needed to produce estimates of  $\dot{\mathbf{l}}$ . This has the unfortunate effect of compounding error, as numeric differentiation is not inherently a "smoothing" operation. Thus a prediction of  $\mathbf{l}$  that is as accurate as possible is needed, likely by performing polynomial smoothing over many observations before

attempting to solve for  $\mathbf{I}$ 's time derivatives. This requires more observations, which is the inherent weakness of Laplace's method. With small amount of data (such as the 3 observations minimally required), the ephemerides produced are not accurate enough for practical use. Given large amounts of data, however, this technique produces highly accurate RSO ephemerides (Taff suggests elements within 0.01% of truth as common), making Laplace's method a common choice for initial orbit determination even at present day. [53]

### 2.2.3.2 *Gauss' Method.*

At the beginning of the 19th century the astronomer Giuseppe Piazzi discovered the asteroid Ceres via telescope. It disappeared behind the sun shortly thereafter, leaving very few recorded observations of its position. From this limited angle-only data, in 1809, Carl Friedrich Gauss was able to calculate Ceres' orbit. In 1888 George James Gibbs refined Gauss' technique into its present day form. The main difference between this approach, and that used by Laplace is that here the observations times are bridged using the unperturbed equations of motion. This is possible by using a power series expansion. Gauss demonstrated the technique using up to the  $t^2$  term, and Gibbs showed how to expand it to the  $t^4$  term.

The most glaring problem with this technique is its convergence conditions. It will converge provided the time between observations is less than  $\frac{P\tau}{2\pi}$  where  $P$  is the orbital period and  $\tau$  is a function of the eccentricity and mean anomaly of the object at that time instant. This time could potentially be quite short, and more problematically, there would be no way to know the value required to converge without first obtaining half of the orbital parameters. In practice, Gauss' method is fairly resistant to sensor noise and provides highly accurate interpolation between observations. However, it is less accurate than desired when extrapolating outside the observation time interval, and may diverge unless a priori knowledge of the orbital parameters is used in obtaining the observations.

This makes the technique less than ideal, though it does form the theoretical baseline for many modern approaches. [53]

#### **2.2.3.3 *Partial Knowledge Methods.***

In real-world applications of initial orbit determination, more information is often available due to the nature of the problem. For instance, you may know that you are looking at a planet, or an asteroid, or a GEO object. Whatever the case is, those bits of information imply a surprising amount about a given RSO's orbital parameters. Then when calculating the ephemeris via, say, Laplace's or Gauss' method, one might also constrain eccentricity to be near 1, semi-major axis to be larger than 40,000 km, etc. This allows the aforementioned techniques to converge on solutions faster and with greater accuracy. [53]

#### **2.2.3.4 *Admissible Regions.***

Because of the proliferation of space objects, it is not always possible to obtain many sequential observations of a given RSO. In these cases “determining” the RSO's orbit, in the traditional sense, may be impossible. The community has dealt with this limitation in a few ways. One of them is a more “fuzzy” interpretation of the RSO's ephemeris. They instead think of a determined orbit as a probability distribution of potential object locations and velocities. This stochastic approach to the problem is addressed by Tommei as a construct called an *admissible region* [55]. Here input is thought of as an attributable vector  $(\alpha, \delta, \dot{\alpha}, \dot{\delta})$  which will then be used to obtain a region of the parameter space where a satellite resides. This region can be further constrained by making assumptions regarding the satellite's orbit, such as the orbit being closed, or in our case closed and geosynchronous. Still, these regions are often larger than astronomers prefer. Milani has specifically applied the admissible region methodology to asteroid tracking, demonstrating the effectiveness of this methodology for orbit determination even with very few observations over a short timespan [38].

Maruskin’s approach [35] instead supposes that orbit determination is only possible if more observations with sufficiently large angular spacing can be obtained. Then the problem is to determine how one would associate optical observations of the same RSO to one another. The SSA community refers to this binning of observations by observed object as correlation (not to be confused with the statistical meaning of the term). Maruskin also refines the admissible region with stricter constraints. The distribution of potential satellite states, as given by Tommei, resulted in about half of the potential RSO states crashing into the Earth when propagated forward several hours, indicating a large portion of the “admissible” region was not actually admissible, assuming the satellite in question is intended to survive more than a few more hours. Maruskin’s technique thus yields a more precise admissible region for a given RSO.

#### ***2.2.4 Observation and Track Association.***

With the previous information regarding GEO RSO tracking and orbit determination in mind, a new problem becomes evident. Many observations of a given RSO must be aggregated before they serve any realistic tracking purposes. This necessitates either collecting observations over long periods of continual observations, or more likely revisiting the RSOs while observing others in the interim. A trade off regarding observation frequency and latency of the catalog arises, which is the subject of the following section on sensor scheduling. Here the concern is how one can determine which exiting tracks new observations ought to correspond to, or which observations collectively correspond to the same untracked RSO. This problem is termed association, with the former being known as observation-to-track association and the latter observation-to-observation association. Association is, in fact, the principle concern of this research and in many ways can be thought of as a prerequisite to orbit determination, RSO tracking, and indeed SSA in general.

The community has approached association in a manner of ways. Most obviously one could simply associate a new observation to the closest cataloged RSO to that observation, where closest here means shortest distance between the RSO and the observation in the sensor's own azimuth and elevation topocentric coordinate system. In fact, though this may seem naive, in many cases this will yield correct associations. In the cases where it does not, however, more sophisticated techniques are required. The launch of new RSOs, or the creation of debris after a collision are of most concern to the community.

Multiple hypothesis tracking (MHT), a technique proposed by Singh, is perhaps the most popular system scale association algorithm available to the SSA community. It is system scale in that it operates on data from many different sensors, and indeed many different kinds of sensors such as radar as well as electro-optical telescopes. It is capable of associating tracks to many different orbital regimes, and can do so efficiently. It works by approaching association as a combinatorial optimization problem where the value being optimized is a likelihood-based score on potential data sequences (called hypotheses). These different data sequences model the entire life cycle of the RSO under consideration, from birth (orbit initiation), to survival (orbit continuation), and to death (orbit termination). This technique has been demonstrated to operate on tens of thousands of uncorrelated tracks and observations accurately and in real time [50]. Siminski's work takes the traditional MHT process and improves upon it by re-framing it as an optimization problem over a given admissible region. Then taking the characteristics of the loss function into account, an effective solution can be found more quickly without a notable change in the rate of erroneous object detections [49], [48], [47]. Stauch improves upon a major limitation of the constrained admissible region multiple hypothesis filter technique. In cases with large degrees of measurement uncertainty the aforementioned technique produces less than desirable accuracy. To combat this a joint probability data association algorithm is incorporated into the multiple hypothesis filter to account for the multi-sensor, multi-target

nature of the data association problem. Additionally, Stauch includes a data smoothing algorithm to mitigate uncertainties on prior observation as additional observations of a given RSO are obtained [51]. Hussein proposes the use of probability distributions and an information theoretic divergence metric to determine which tracks to associate to one-another. This was compared to techniques based on track covariance, and evaluated via simulation. The divergence metric approach yielded superior results across all considered cases, including multiple orbital regimes [26]. Fujimoto examines the admissible region association and initial orbit determination technique with the addition of a least squares estimator. Of note, many of the tracks cannot be associated correctly, leading this work to conclude that astrometric observation alone may be insufficient to solve the association problem; a claim that will be addressed in the remainder of this document as well [16].

#### ***2.2.5 Sensor Scheduling.***

Also of interest to the community is how to allocate the finite available time of a finite number of sensors to observe as many RSOs as possible. One must, after all, have a collection of observations before any association can be performed. Keeping in mind the vast number of RSOs that must be observed, the heterogeneity of the SSN, and the needs of orbit determination algorithms is a non-trivial problem. Varied approaches and metrics are used in approaching this problem, and some of those are described here.

Petrick approaches sensor scheduling from an optimization perspective. Given a goodness or desirability function for a potential schedule, a simulated annealing optimization algorithm is adapted to produce priority weights on the RSO catalog that yields the most desirable schedule [42]. Miller proposes a sensor scheduling algorithm based on observation density, where each RSO would have an a priori observation density requirement to which the scheduler must adhere [39]. Hill describes an environment called TASMAN (Tasking Autonomous Sensors in a Multiple Application Network) for use in testing electro-optical sensor scheduling algorithms. With this framework they

then demonstrate improved accuracy in catalog ephemeris by exploiting RSO state error covariance [22]. Hill in a further study examined a distributed sensor scheduling algorithm and a fully centralized sensor scheduling algorithm and compared these to a baseline representing the current operational scheduling approach. Significant improvements were noted in both cases, as they both leverage the aforementioned covariance metric to assess observation quality [23]. Hobson proposes an alternate simulation environment to TASMANT based on Matlab and leveraging the parallel computing toolbox of that language. Though this environment is lower fidelity than TASMANT, Hobson notes that it is more adaptable and demonstrates a scheduling algorithm that accounts for both electro-optical sensors as well as radars [24]. Newman evaluates a collection of scheduling techniques by comparing expected entropic information gained due to given observations. Such a technique is intended to work in tandem with the MHT association system discussed previously. Newman notes that particle swarm optimization (PSO) combined with an evolutionary algorithm (EA) produced the best results, where PSO, EA, and simultaneous perturbation and stochastic approximation (SPSA) were considered [40]. Greve advances Newman's work by considering a larger scale scheduling problem. Their approach, termed Evolutionary Algorithm Tasker (EAT), is able to operate on a large number of sensors (8) and RSOs (20000), though slightly fewer RSOs that are given tracked assignments are successfully tracked than with the currently fielded technique, named SP Tasker [19].

Sensor scheduling is a major concern for nearly every aspect of SSA. In order to track RSOs you must have a collection of observations to start with. In the context of a simulation, though one might desire observations be produced for all RSOs at a regular interval throughout an entire scenario, in practice this might not happen. For this research, sensor scheduling concerns were considered when selecting the value for minimum time between observations.

### 2.2.6 *Photometry.*

There are two drawbacks of tracking objects using only astrometric data. First, without range information there is a significant amount of uncertainty in the object's true location. This uncertainty can be mitigated only partially by increasing the number of observations. The second, and more systemic problem, is that there is no way to be sure which object is currently being observed. In fact, it could be quite difficult to differentiate RSOs with orbits that either cross or place the RSOs consistently near one another. In the extreme case, in which satellites conduct rendezvous and proximity operations (RPO), once they separate, angles-only data is not sufficient to determine which track belongs to which satellite.

Optical observations contain more information than angles-only location, however. The intensity of light over time, also known as the photometric light curve or just photometry, is the information captured and then characterized by the sensor. Though that can certainly be processed into angle estimates, it possesses more information. Kuroskai demonstrates spinning satellites that yield recognizable light curves, in that their spin-rates can clearly be determined with a transformation of the light curve signal to the frequency domain (i.e. FFT) [32]. Rodriguez specifically examines assumptions of object albedo, or material composition. They discovered that both composition and shape drive the form of the observed light curve, and attempted to construct a preliminary taxonomy of their photometric signatures. Ultimately the goal is to characterize the kinds of light curves one would expect from objects of a variety of compositions, shapes, and sizes [43].

Most obviously, one could extract periodic elements, average brightness, and a rough estimate of object size based on the raw light curve. Theoretically, the photometric light curve is affected by a number of factors, to include sun-object angle, object-sensor angle, and the object's incident illumination. Most importantly, the light curve is a function of the object's attitude and body shape. The process of estimating either of these is called light curve inversion, and has been studied extensively as it relates to determining the shape

of asteroids [41] [28]. Hall specifically approaches the problem of characterizing attitude and body shape for an arbitrary object from its light curve, and comes to the conclusion that they should be extracted separately in order to obtain the best possible accuracy and tractability. He also notes that light curve inversion is based on the assumption that the RSO's geometry is convex. This is a generally accepted assumption when the process is applied to asteroids, though it may not be appropriate for man-made satellites. That being said, the impact of the object convexity assumption is unknown, though it limits the current practical application of light curve inversion [20].

In this research, photometry is referred to often. With everything said thus far in mind, it would be reasonable to expect photometry be a way to map observations to their parent RSO. The above caveats, however, temper expectations somewhat. If light curve inversion could be performed, then photometric-based association would be a solved problem. Though even though inversion likely isn't possible, it may be that sufficient geometric information can still be extracted to perform said association.

#### ***2.2.7 Satellite Maneuvers.***

Understanding RSO behavior necessitates understanding the way in which they operate. In particular, it is important to understand the way that satellites keep station, or maneuver to a new station. Few operational satellite's simply drift, especially in GEO where space is quite valuable. In either case the satellite will accelerate or "burn" to counteract drift, or else to place itself into a new desired orbit. The case of satellite rendezvous is an extreme case of this where virtually no drift can be tolerated. The following sections discuss how said maneuvers are accomplished.

##### ***2.2.7.1 Station Keeping Maneuvers.***

A distinction needs to be drawn between two classes of geosynchronous orbits. Satellites in a geostationary orbit (GSO) that nearly remain stationary over the same location on the Earth, and are allocated portions of the GEO belt (called "slots"

or “stations”). The remainder of GEO satellites are somewhat inclined; they will approximately retrace their track every day, though that track looks more like a “figure-8” or oval over the Earth’s surface. It does not make sense to talk about a station or slot for these satellites.

In order to comply with international regulations, GSO objects must maintain their station to within a certain tolerance. Stations are assigned by the International Telecommunications Union (ITU) where the only stated requirement is that the satellites do not interfere with the signal transmission or reception of one-another. In practice, this often amounts to an assigned  $1^\circ - 2^\circ$  longitude slot, though most antenna require the satellite stay within a  $0.1^\circ$  latitude-longitude box. Fortunately, this means the ITU can pack satellites more densely where the demand is high, such as over Europe or Asia where many countries need slots<sup>1</sup>.

Chu, et al. discuss a potential schedule of maneuvers that allows satellites to maintain their station by making daily small thrust maneuvers [13]. There are advantages of smaller, more regular maneuvers, namely that they cause less disruption to satellite instruments and thus increase sensor availability. Satellites drift in inclination, eccentricity, and longitude. Inclination drift is mainly due to the gravity of the Sun and Moon. This results in about  $1^\circ$  of inclination drift per year if not counteracted with a north/south burn. Eccentricity drift is mainly due to uneven solar pressure, causing either an RSO’s apogee or perigee to differ from that of a circular orbit. This is largely unavoidable, as satellite attitude is often a function of mission, and thus periodic burns to maintain a circular orbit are necessary. Finally, longitude drift occurs for two main reasons: accumulation of angular momentum due to solar pressure and the non-spherical geopotential (i.e. gravity) field of the Earth. In order of severity, inclination drift requires the most thrust, followed by angular momentum

---

<sup>1</sup>Though some foreign powers have declared sovereignty over the portion of geosynchronous orbit above their country, the claims have no basis in international law and the ITU remains the authoritative source on GSO station allocation [1].

dumps to correct for eccentricity drift, and then lastly by east-west drift compensation. A notional station-keeping maneuver schedule is given below in Table 2.1 [13].

Table 2.1: Sample maneuver schedule to minimize individual maneuver magnitudes [13].

|                            | thruster axis | magnitude $\delta v$ (m/s) | period (days) | daily $\delta v$ (m/s) |
|----------------------------|---------------|----------------------------|---------------|------------------------|
| east-west stationkeeping   | tangential    | 0.27 – 0.29                | ~ 80          | 0.0035                 |
| momentum dump              | any           | 0.01                       | 1             | 0.01                   |
| north-south stationkeeping | out-of-plane  | 45 – 50                    | 365           | 0.13                   |

### 2.2.7.2 *Rendezvous and Proximity Maneuvers.*

Satellite rendezvous can naturally be divided into two questions. First, how does a satellite go from its initial orbit to the orbit of a different RSO? And second, once it reaches that new orbit, what does the satellite do? For the purposes of this research we can even consider a satellite moving to a new unoccupied orbit as a sort of rendezvous, just without an object present at the destination. These two questions are addressed in turn.

In order for a satellite to go from one orbit to another, an understanding must first be established as to what this means. An orbit is a 6 dimensional state, meaning that it requires 6 independent quantities to accurately characterize. Position in space only requires 3 quantities, meaning that being at the correct location is insufficient. One must also be moving at the correct velocity. Together position and velocity represent 6 quantities in space. This means that, intuitively a satellite must first cross over a desired position in its new orbit and then change its velocity vector to that of the desired orbit. In fact, given infinite fuel, this would be the easiest way to achieve a different orbit.

In practice, fuel is a limited resource for any satellite. A more constrained and prescriptive method exists, however, known as a Hohmann transfer orbit [10]. A Hohmann

transfer uses the smallest amount of energy possible to move between two circular orbits in the same plane but of different radii. This makes it the preferred orbit transfer technique in many cases so as to maximize mission payload. It works by having the satellite in question accelerate at either apoapsis or periapsis, the highest or lowest point of the orbit, respectively. When a satellite accelerates along its velocity vector this causes the altitude of the opposite side of its orbit (from the satellite's current position) to increase. Thus the satellite can change the altitude of the opposite side of its orbit to match that of the second orbit. Once the satellite has reached that raised point on the opposite side of the orbit, the satellite accelerates again so that its lowest point (the point in the orbit where it formerly accelerated) is raised to the desired altitude. A diagram of this procedure can be seen in Figure 2.2.

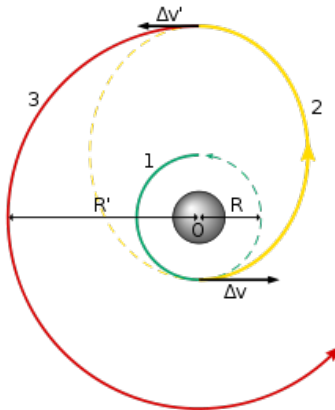


Figure 2.2: Graphical depiction of a Hohmann transfer orbit [33].

The only remaining complexity is how to reach that new position at the correct time. For instance, when rendezvousing with the ISS, one would need to not only reach the same circular LEO orbit, but also be nearly co-located in that orbit. In a Hohmann transfer this is accomplished by timing the initial acceleration. Because the object's current orbit has a different radii than the desired orbit, there will be a non-zero relative angular velocity. Then

the satellite (or manned spacecraft in the case of the ISS) need only wait until the desired configuration, given the mission permits the necessary time.

In practice, this last constraint motivates the reason operational satellites may not use a Hohmann transfer. If the mission does not permit waiting for the optimal configuration, then additional fuel will be required. This will decrease the effective lifespan of the satellite in order to achieve an earlier rendezvous. To what extent this trade-off is acceptable is a function of the mission and satellite value.

In the most apparent case, after rendezvous the objects physically dock or join themselves together. In SSA it is more typical for the satellites to stay within visual range and move relative to one-another over some time period. This could be done as before docking in order to find an appropriate location to grapple with a non-cooperative target such as debris. It could also be done to survey an unknown satellite. After examining the unknown satellite from a collection of angles one would obtain a detailed understanding of that satellite's geometry. There are a few techniques to accomplish this, such as natural motion circumnavigation or tear-drop orbits [17] [60]. In general there is always a trade-off between how quickly information is gained and how much fuel is spent in the process.

In terms of this research, it is necessary to model RSO maneuvers so as to obtain representative photometric observations. For all maneuvers used in this study a Hohmann transfer is leveraged for standardization. For any RSO keeping station, a form of the schedule given in Table 2.1 will be used. Though the main advantage of that schedule is to minimize equipment vibrations, which is inconsequential to this study, said schedule does offer a standard and repeatable method for an RSO to maintain station.

### **2.3 Machine Learning**

In addition to SSA, this research draws upon machine learning to conduct the autonomous association of RSO observations with their labels. Here an overview and introduction to the machine learning lexicon is given. Afterwards there is a discussion

of classification specifically. Finally this section concludes with artificial neural networks (ANNs) and an important variant of ANNs: convolutional neural networks.

### **2.3.1 Overview and Definitions.**

Machine learning is concerned with developing algorithms to map inputs (called observations) to outputs (called labels). Before approaching the problem, it is assumed that some relationship between our inputs and outputs exists, but that we do not explicitly know it or even understand it particularly well. The machine learning process itself is intended to extract that relationship autonomously. The algorithms (hereafter referred to as models) are intended to search the space of potential relationships and choose the “best” one. Determining which relationship is, in fact, best is difficult.

This is accomplished by providing the model with a training set. The model examines the entire training set and updates itself, then repeats that process. One examination of the entire training set is termed an *epoch*. Typically the training set is viewed in smaller pieces, and smaller updates made after each of those subsets. These are called batches, and the batch size the number of observations viewed at a time before updating model weights. Training can be terminated after a preset number of epochs, or if some early stopping criteria is met. Often a patience value is set, meaning that if a pre-ordained number of epochs pass without model improvement, training will terminate regardless of how many epochs remain. This can be a tremendous time-saving technique, especially for models with very long training times.

Naively one might want a model that perfectly labels every training sample, but in practice these models may perform poorly. This is because noise is present in nearly all real world phenomena, and a model may fit to that noise if given the chance. Such a model will fail to generalize beyond training data, making it useless for all practical purposes. The term assigned to the phenomenon where a model fits to noise and thus fails to generalize beyond its training data is ‘overfitting’. To detect this, machine learning

researchers usually introduce separate sets of data formed by randomly partitioning the initial dataset. The training set is the one in which the model is provided the labels in order to search for an ideal fit. The validation set is provided to the model without labels, so that the performance on that set is a reliable indicator of model generalization. Finally, if multiple models are considered, the one selected as best (presumably because it had the best validation performance) should be evaluated on a held-out testing set to get a realistic estimation of its generalization, as the validation performance is biased by the use of the validation set in the model selection process.

Often a procedure known as  $k$ -fold cross-validation is employed when evaluating models. To do this, the data is split into  $k$  random partitions called folds.  $k - 1$  of those are used for training (and validation, if the user desires). The last fold is held out until the end of training. The model's performance (classification accuracy in the case of classification, mean squared error in the case of regression) on that fold is recorded. This procedure is then repeated with a new model with a different fold taking the role of held-out testing set. This means that the machine learning algorithm itself is the same for each fold, though the weights are trained from scratch every time. In total  $k$  testing performance estimate are produced from this process.

### **2.3.2 *Classification.***

Models broadly fit into two categories based on the kind of label they predict. Classifiers predict categorical labels, and regressors predict continuous numeric labels. For instance, predicting the make of a car is considered classification, whereas predicting engine displacement is considered regression. Classification is of particular interest here, as RSO identity is a categorical variable. The simplest case is the two-class problem in which the model provides a Boolean response to each observation. Multi-class problems can be viewed as simple extensions of the two-class problem by viewing an observation's label as a "one-hot" vector. Instead of making the label a natural number

$i$  indicating that an observation belongs to class  $i$ , the label is instead a Boolean vector (e.g.  $(0, 0, \dots, 0, 1, 0, \dots, 0)$ ) containing a single 1 in the  $i$ th position and  $n - 1$  0's in the remaining positions, where  $n$  is the number of classes. This is equivalent to simultaneously answering  $n$  two-class problems, where problem  $i$  is to determine if an observation is either class  $i$ , or any class  $j$  where  $j \neq i$  [52].

Notice that in an  $n$  class problem, a totally random labeling would be expected to yield  $100 \cdot \frac{1}{n}\%$  accuracy. So in a 2 class problem, even a coin-flip based approach should give at least 50% accuracy. This gives readers a milestone from which to measure classifier performance. Performing significantly better than random, in multi-class problems, doesn't require near 100% accuracy. And one wouldn't expect a classifier to perform significantly worse than this threshold.

A model with sufficient capacity can always learn the training set well enough to perfectly classify each training observation. This is an extreme form of overfitting that occurs in classification called “shattering” the dataset. To avoid this, techniques need to be employed that penalize unnecessary complexity within the model. These techniques are collectively referred to as regularization.

### **2.3.3 Artificial Neural Networks.**

Artificial neural networks are a particular family of models often referenced in modern (or deep) machine learning. The fundamental element of an ANN is a neuron, motivated by the biological cell of the same name. The inputs to the virtual neuron are each individually multiplied by “synapse” weights and the results summed. This value is given as input to a non-linear activation function that produces the neuron's output activation. The synapse weights are the trainable parameters within the model, and the non-linear activation function permits compositions of neurons to produce complex behavior, rather than behavior that could be obtained from a single neuron. Theoretically, provided one

uses non-linear activation functions, ANNs could replicate the mapping of any training set, provided the network contained a sufficient number of neurons [18].

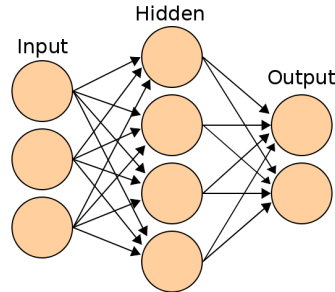


Figure 2.3: Artificial neural network with one hidden layer [15].

ANNs are typically structured as layers of neurons in which the outputs of one layer are provided as the inputs to the neurons of the subsequent layer (see Figure 2.3). This paradigm is mainly convention, though it is easy to understand and better enables communication of results as it is widely leveraged. The process of training these networks is based on propagating credit/blame backwards through the network to determine which neurons are most in need of change. Further information on the training process can be found in many sources describing various ANN optimizers, such as SGD (stochastic gradient decent), RMSProp, or Adam [30].

When an ANN is used for classification, its output layer contains a neuron for each class with a *softmax* activation function. *Softmax* is a multi-dimensional extension of the logistic, or sigmoid, function. Because of the way *softmax* is defined, the activation value of each output layer neuron can be interpreted as the probability of membership within the corresponding class [18].

At the time of this writing, the state-of-the-art approach to ANN design is to limit the width of each intermediate layer, but to include many layers (hence “deep”) [18]. Such networks generally have larger capacity (number of trainable weights) than needed,

and thus tend to overfit. Several regularization techniques exist to combat this. Batch normalization serves to counteract the impact of updating layers sequentially. As an update to an early layer can have large impacts on later layers that still need to be updated, the impact can be quite severe [18]. More commonly used in deep learning, dropout randomly removes a percentage of neurons during each training batch. This is achieved by examining the output activation of each neuron and stochastically, at some preset frequency, fixing those neuron's activations to 0 [18]. The network then must learn to cope even with individual neurons unable to respond. This not only mitigates overfitting, but also allows the network to develop more versatility and redundancy at the expense of slower training.

### ***2.3.4 Convolutional Neural Networks.***

#### ***2.3.4.1 Overview.***

At present day one of the most well known artificial intelligence problems is image classification. This is mainly because there was previously no effective way to approach this problem without a large number of man-hours of feature engineering. Neural networks offered a way to learn the mapping from pixels to correct class autonomously, and have been successful in that endeavor. However to achieve that success a major obstacle had to first be overcome. That is that the sheer number of pixels of an image make treating each pixel as an individual feature impractical. Instead some degree of weight sharing is necessary, otherwise one cannot expect the training process to conduct even a reasonable search over the image's feature space. Convolutional neural networks provide exactly that. They are mainly exercised through two unique layers: convolution layers and pooling layers. These are briefly discussed, followed by an early example of deep convolutional neural networks.

#### ***2.3.4.2 Convolution and Pooling.***

Convolution is a signal processing technique wherein a small kernel is applied to some source signal [18]. The product of that kernel with the original signal is recorded as that kernel is moved in time along the length of the signal. Though images don't have a time

component, the kernel is instead moved spatially. The output of such a convolution is effectively a new image. Convolution layers, as the name implies, apply a collection of kernels to the input image. These kernels, sometimes called filters, are usually very small; either  $2 \times 2$  or  $3 \times 3$  kernels are popular in image classification. The trainable weights of the layer are then the values of each of those kernels. This encapsulates the weight-sharing aspect of convolutional neural networks: though thousands of pixels are present in an image, a single filter can contain as few as 4 trainable weights (other than the bias weights). Then, as the results of a convolutional layer are simply images, one can think of these results as intermediate representations of the input image. In fact, one could apply a convolution to those in the exact same way. Such a topology is depicted in Figure 2.4.

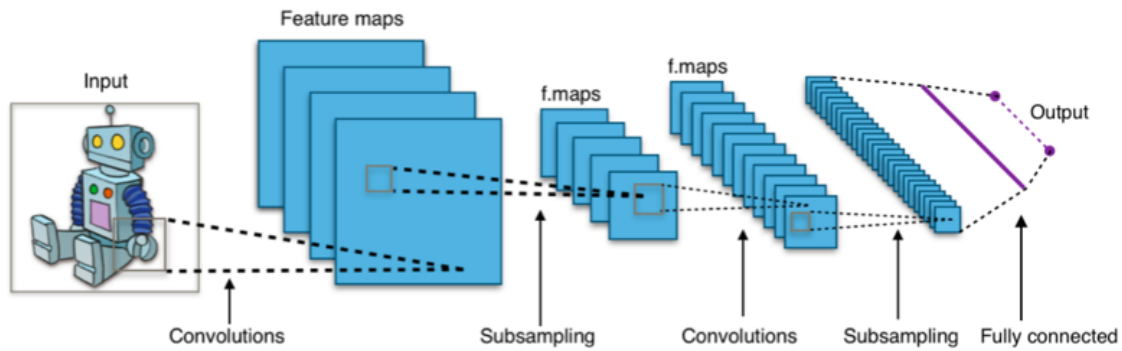


Figure 2.4: Topology of a typical convolutional neural network. Large squares correspond to intermediate representations. Small squares correspond to the following layer’s kernel to be convolved with the current layer. Squares are stacked to indicate the individual filters learned (sometimes called depth). Subsampling is synonymous with pooling [8].

Pooling is a regularization technique, meaning it serves to decrease the size of the feature space so as to combat overfitting [18]. Pooling is typically accomplished via the maximum, minimum, or average function. A pooling layer will have some sample rate, and will combine pixels of its inputs using the aforementioned function at whatever rate

is desired. Thus if the sample rate is 2, the image produced will be half the size of the one input to the pooling layer. Such layers are usually spaced throughout a convolutional neural network topology as a sort of funnel. At some point in a convolutional topology intended for image classification the image must be flattened, meaning it must become a series of features with no spatial dimension to be provided as input to a fully-connected layer. Pooling is one way to ensure that the image is sufficiently small as to flatten without creating an unmanageable number of weights in the first fully-connected layer.

#### ***2.3.4.3 Deep Convolutional Neural Networks.***

The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is a machine learning competition that drove the innovative techniques that would eventually be called deep learning [44]. It is a 1000 class problem with roughly 1.2 million training images, 50000 validation images, and 150000 testing images. All this to say that it is a hard problem. In 2010 the dominant topology was one called AlexNet which achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry [31]. It accomplished this by introducing several innovations that have since become commonplace. First, the convolutional net used had many layers: 5 convolutional layers, 2 fully-connected layers, and then the output layer. This is why the term “deep” is used to refer to these topologies. For regularization, though pooling was used, dropout was also introduced. Finally, the training of this network was accelerated with a GPU. All of these together amounted to AlexNet effectively setting the standard for deep learning approaches over the following years.

#### ***2.3.5 Time Series Classification.***

Though the application of deep neural networks to RSO photometry seems to be a novel concept, the use of time series data in classification is not a new concept. And as photometry is essentially just time series data, many existing works are, from a machine learning standpoint, quite similar. For instance, in 2012 Abdel-Hamid’s research used

convolutional neural networks for human speech recognition and exceeded state-of-the-art accuracy with that technique [7]. In 2015 Wang used exiting computer vision techniques to approach time series classification [57]. This was accomplished by first encoding the time series as an image by using either Gramian angular fields, or Markov transition fields. These results were comparable to exiting techniques. In 2016 Cui approached this same problem by employing a multi-branch convolution net where each branch leverages a different frequency sampling of the original time series data and achieved superior accuracy to state-of-the-art techniques on the UCR database of time series classification problems [14], [11]. In 2017, Wang demonstrated several ANN topologies, namely multilayer perceptrons, convolutional neural nets, and residual networks, that outperformed previous techniques on the UCR database [58]. Most real world variants of time series classification require multi-channel data be considered. To this end, Zheng demonstrated a deep convolutional neural network architecture that yields comparable accuracy to state-of-the-art techniques (1 nearest-neighbor with dynamic time warping), though at a considerably improved runtime [62]. Multi-channel time series is also the input data in Yang’s research where human activities are predicted from body-worn inertial sensors by using deep convolutional neural networks [61]. Hefron further demonstrates a hybridized multi-path convolutional and recurrent neural network for use in modeling the relationship between human electroencephalograph (EEG) data and cognitive workload, and noted superiority of that technique to existing topologies [21].

## **2.4 Summary**

Here a brief discussion of both SSA and machine learning were given. The complexity involved in GEO track association motivates the approach used in this study. In addition, the treatment of machine learning introduced that domain’s lexicon. More importantly perhaps, that section introduced the algorithms used to address these kinds of problems.

With this all in mind, the following chapter addresses the methodology used to apply deep neural networks to the RSO association problem.

### **III. Approach and Methodology**

The objective of this study is to demonstrate the feasibility of association GEO RSO observations using deep neural networks. To that end, this research hypothesizes that deep neural networks (DNNs) trained on photometry can be applied to GEO SSA to address current UCO and UCT association ambiguities. That hypothesis is address through three research questions:

- Are there any cases where a DNN can use photometry to correctly associate observations to the corresponding RSO?
- Can a DNN use photometry to associate observations of RSOs even if those RSOs are closely spaced?
- Can a DNN use photometry to associate observations of RSOs even if those RSOs are maneuvering?

An experiment is conducted to assess each of these questions. The discussion begins with the items that all three experiments share in common: an overview of terminology, the dataset generation, and the application of DNNs. Afterwards the experiments are discussed individually.

#### **3.1 Overview**

For each experiment described here a dataset is first generated representing the appropriate phenomena via AGI's Systems Tool Kit (STK). To do this ephemeris are first created for the RSOs in question, then the photometric aspects are modeled and a dataset of observations is generated. Afterwards, a collection of machine learning models will be trained and assessed on that dataset using a cluster-based experiment framework. A diagram of this process can be found in Figure 3.1.

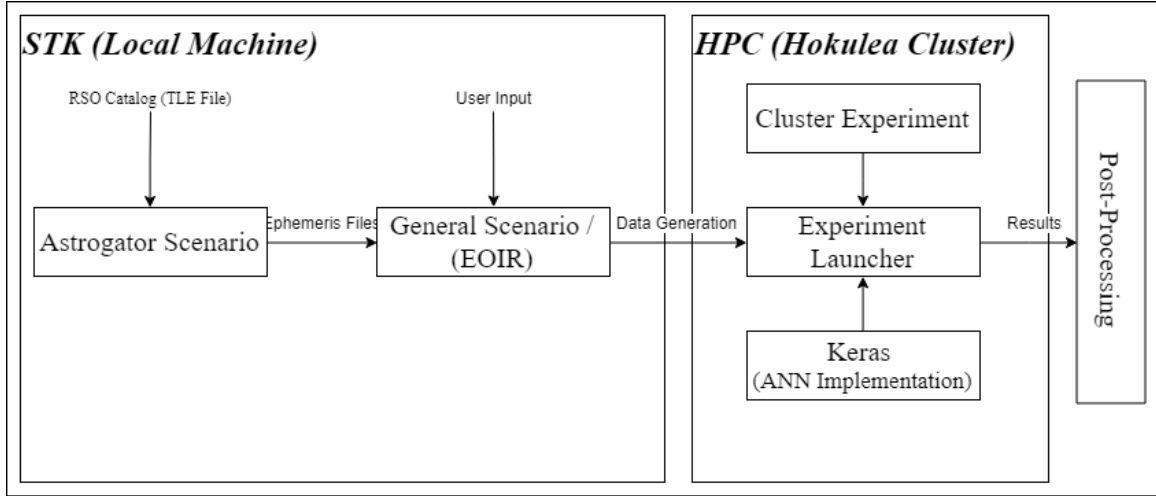


Figure 3.1: Diagram of process used in this research. From left to right ephemeris are created for the experiment in question, then photometric aspects are modeled and observations produced. Afterwards various cluster resources are used to evaluate a collection of ANNs on those observations, and the results of the experiment are produced using those resources.

The aforementioned machine learning models will have been presented with thousands of observations of these RSOs throughout the training process. An appropriate concern would be that, under real world constraints, should such a body of labeled observations exist then this problem would be trivial. However the photometric signature of the RSO corresponding to that UCT might have been observed before as well, provided that RSO has ever been tracked previously. In this case it may be possible to construct a system that correlates that UCT by noting a similar photometric signature to that of a known RSO. Furthermore it is important that association of the observations generated here be considered as individual and isolated decisions. Else one would need to model the impact of correcting mis-associations on the remainder of the scenario. This would undermine the autonomous nature of the technique in question, so it is not considered.

### 3.2 Dataset Generation

A myriad of tasks are necessary in order to create a dataset sufficiently representative of real-world phenomenology as to be useful. As was stated previously, STK will be used to model that phenomenology. Starting with a catalog of GEO RSOs taken from Space-Track.org on 31 July 2017 [6], this catalog is then restricted to 8 specific RSOs in the Maui GEODSS site’s field of view (FOV) with RSOs at relatively different stations. These RSOs are JCSAT 16, INTELSAT 907, GOES 13, AMC-16, GOES 16, DIRECTV 11, MEXSAT 3, and GALAXY 15 in the aforementioned Space-Track catalog. The scenario used started on 1 Jan 2019 at 1200 UTC and lasted until 1 Jan 2020 at 1200 UTC.

STK allows one to specify an RSO’s photometric properties. That is, one can specify RSO material, attitude profile, and geometry. Regarding materials used, STK models the entire surface of the RSO as being of the same material. This amounts to the spectral response, meaning the wavelengths of light absorbed vs reflected, for that RSO. The spectral response curves themselves are internal to STK and are based on NASA Reference Publication 1121 [56]. The attitude profiles considered in this research are nadir alignment with orbit normal constraint, and nadir alignment with Sun constraint. The former aligns the RSO  $z$  axis to the nadir direction (opposite the geocentric position vector) and constrains the RSO’s  $x$  axis to be in the direction of the orbit normal vector (example in Figure 3.2). Similarly the second profile constrains the RSO’s  $z$  axis to point in the nadir direction and the RSO’s  $x$  axis to point towards the Sun (example in Figure 3.3). Finally, the shapes used are the GEOComm and LEOImaging shapes, which are internal to STK.

It is also necessary to model a sensor that will generate the dataset observations. In particular, photometry is highly sensitive to the observer’s CCD, so the choice of sensor is important. Here the GEODSS telescopes and CCD(s) were used. The operational GEODSS sensors are located at three sites: Maui, HI ( $20.71^\circ$  N by  $156.26^\circ$  W), Diego Garcia ( $7.41^\circ$  S by  $72.45^\circ$  E), and Socorro, NM ( $32.82^\circ$  N by  $106.66^\circ$  W). This study focuses on the

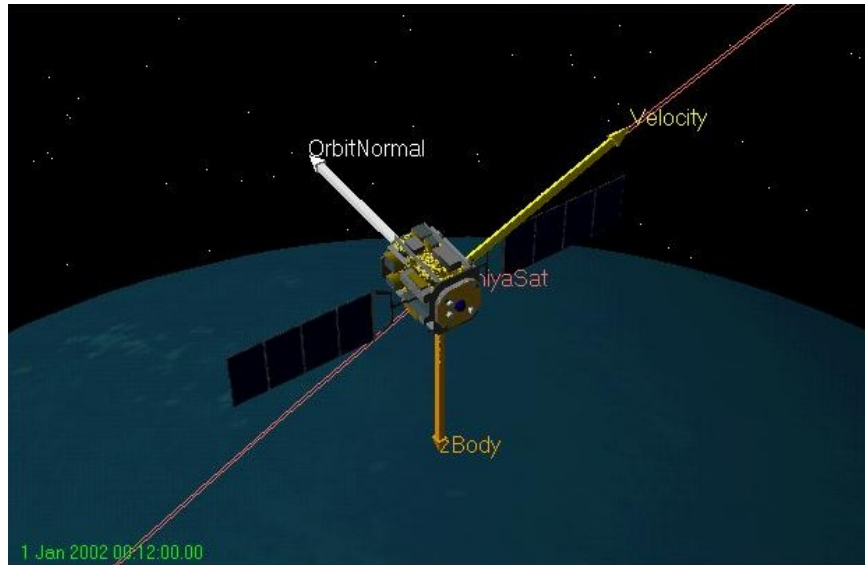


Figure 3.2: Graphical depiction of the “Nadir alignment with orbit normal constraint” attitude profile. From AGI’s website at: <http://help.agi.com/stk/index.htm>

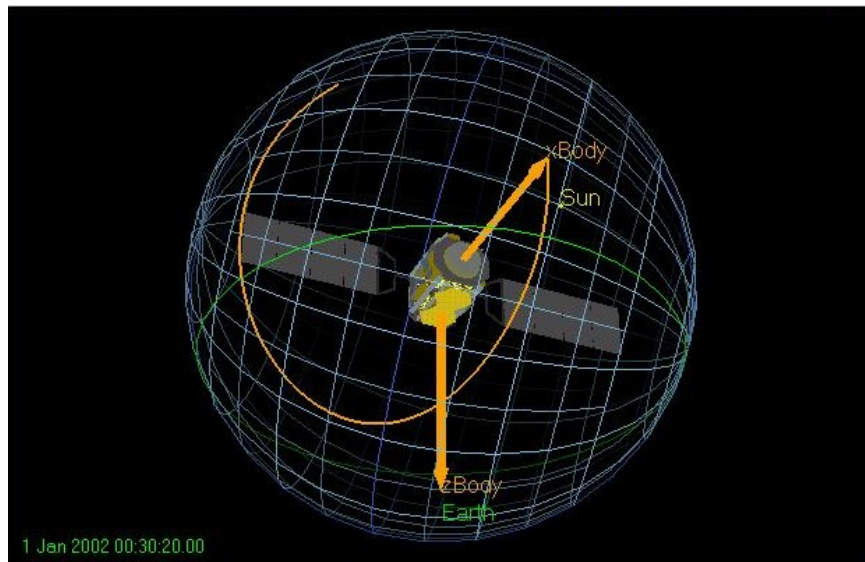


Figure 3.3: Graphical depiction of the “Nadir alignment with sun constraint” attitude profile. From AGI’s website at: <http://help.agi.com/stk/index.htm>

Maui GEODSS site specifically. Regarding the GEO RSO observations, each GEODSS telescope employs two CCDs: one for detection and one for space object identification (SOI). In this research's lexicon, this would mean one CCD is devoted to astrometry and one is devoted to photometry. As only photometry is being used in this study, only the SOI CCD is modeled. The photometric observations are generated from STK using the EOIR plug-in taking into account the object properties and attitude as well as the GEODSS sensor. The generated photometry only consists of a single channel (from  $0.4 - 0.9\mu\text{m}$ ). The remaining sensor parameters are intended to model GEODSS based on the description given by [5]. The parameters chosen can be found in Table ??.

Table 3.1: Table of values used in STK model of GEODSS space object identification CCD [5].

| Parameter Name           | Value Used | Parameter Name       | Value Used |
|--------------------------|------------|----------------------|------------|
| DarkCurrent              | 6.0        | LowBandEdgeWL        | 0.4        |
| DetectorFullWellCapacity | 160000.0   | QEMode               | 0          |
| DynamicRangeMode         | True       | QEValue              | 0.3        |
| OpticalInputMode         | 1          | QuantizationMode     | 0          |
| EntrancePDia             | 101.0      | RadParamLevel        | 1          |
| Fnumber                  | 2.15       | ReadNoise            | 12.0       |
| HighBandEdgeWL           | 0.9        | SaturationMode       | 0          |
| HorizontalHalfAngle      | 1.23       | SimulateQuantization | True       |
| HorizontalPixels         | 32.0       | SpatialInputMode     | 0          |
| IntegrationTime          | 0.01       | VerticalHalfAngle    | 1.61       |
| VerticalPixels           | 32.0       |                      |            |

The definition of observations used thus far has been somewhat loose, and a more concrete definition is necessary. An observation in this study represents a 6 second light streak of the RSO. We take angle measurements at both the start and end of the streak, and photometry throughout. More rigorously, we define a single observation (starting at time  $t_0$  and ending at time  $t_0 + 6$ ) as the tuple  $(\alpha_1, \delta_1, \alpha_2, \delta_2, \sigma)$  where  $\alpha_1, \delta_1$  are the measured angular positions of the RSO at the start of the observation ( $t = t_0$ ), and likewise  $\alpha_2, \delta_2$  are the angular positions at the end of the observation ( $t = t_0 + 6$ ). Note that the coordinate frame in particular does not matter, and could be local azimuth and elevation, right ascension and declination, or even geodetic (e.g. latitude and longitude). For the photometric portion of the observation let  $\sigma$  be defined by  $\sigma = \{(t, I(t)) : t = 0.0, 0.001, \dots, 5.999, 6.0\}$  where  $I(t)$  is the observed irradiance of the RSO at time  $t + t_0$ . We call  $\sigma$  the *light curve* of the RSO over this time period. An example plot of a light curve can be found in Figure 3.4.

*TODO*

Figure 3.4: Example plot of light curve over a 6 second observation. Signal is sampled at 1000Hz.

For simulation purposes, scheduling concerns are ignored and it is instead assumed that the sensor can observe everything in its field of view simultaneously. A 6 hour minimum time between observations is imposed so that no object is observed unrealistically often. That all being said, the sensor is not free to observe an object at any time. In order to create an observation, the sensor must be in darkness (penumbra or umbra) and the target in sunlight (direct sun). This means that, in practice, the simulation will exhibit revisit rates lower than 1 observation every 6 hours.

As the scenario begins nearly 2 years after these space-track ephemeris are dated, and then that scenario lasts an additional year, SGP4 will certainly not yield a representative

ephemeris. This is because operational GSO satellites must keep a fairly strict station in order to be both in compliance with international law, as well as to be useful. SGP4 will not yield this by default, nor should it. Station keeping burns are necessary, and must be modeled for our observations to be representative. The maneuvers used here are based on the notional burn schedule for station keeping that was given in Table 2.1.

In STK, the Astrogator propagator is used to model the RSO maneuver schedules. Each day will have a corresponding Astrogator target sequence. On day 1 (1 Jan 2018 at 1200) a north-south correction maneuver is conducted. A maneuver is added to the ascending node, and then the following descending node. The sequence examines the latitude a quarter day after both maneuvers and attempts to achieve  $0^\circ$  latitude at those points. A  $0.05^\circ$  tolerance is allowed in either north or south, corresponding to the  $0.1^\circ$  box of a GSO station. This sequence achieves the desired north-south tolerance for the entire year, but the east-west drift and station keeping still needs to be addressed. To do this, the momentum dump and east-west station keeping given in the notional schedule of Chapter 2 are combined. A target sequence is created with maneuver nodes at apoapsis, periapsis, apoapsis again, and then periapsis again. The sequence is told to achieve the desired altitude of apoapsis, eccentricity, and longitude at the time of the 4th maneuver only. The tolerances allowed are 100km, 0.1, and  $0.05^\circ$  respectively. These desired conditions correspond to creating a near circular orbit (to minimize drift) as well as locating one's self over the "correct" longitude slot. Here the longitude slot is selected by taking the mean longitude of the RSO over day 1 according to SGP4. This sequence can take multiple days of simulation time to execute, depending on how far the RSO must move. The additional maneuvers provide more degrees of freedom for Astrogator to use to hopefully achieve convergence. Intuitively, one could think of these 4 maneuvers as performing a Hohmann transfer from the current position to the destination longitude slot. It will be necessary to

repeatedly perform this sequence, as unlike the north-south correction, the satellites will drift from their assigned longitude within only a couple days.

Once the entire sequence has been run and converged, the ephemeris will be output to file. Astrogator unfortunately consumes a vast amount of memory (greater than the 8 gigabytes available on the machine computing the ephemeris), making the inclusion of multiple satellites modeled as described in the previous paragraphs impractical. So we instead disassociate the ephemeris creation from the remainder of the process by reading/writing those ephemeris to file after this step. These files can then be read by the scenario quite efficiently and used to generate a set of observations for each RSO throughout the year. The code used to load these ephemeris files into a scenario can be found in Appendix A. The first week of data is excluded as initially the RSOs may be in an erratic state as they attempt to reach their assigned stations. Manual inspection indicates that behavior appears to stabilize within about a week of scenario time. The code used for dataset generation can be found in Appendix B.

### 3.3 ANN Topology

The machine learning task under consideration is a supervised two-class classification problem. A given model will be trained on photometric observations of two specific RSOs, where the classes the model must predict between are those RSOs. Then each model should map a particular photometric observation to the RSO to which the observation corresponds. The loss function used is the binary cross-entropy function given by the equation,

$$-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

where  $N$  is the number of observations,  $y_i$  the true class label, and  $\hat{y}_i$  the class label predicated by the model. Here class label will be either 0 or 1, which is arbitrarily assigned to the two RSOs under consideration [12]. The main performance metric monitored is accuracy, meaning the percentage of observations mapped to the correct RSO.

To accomplish this task, an appropriate ANN topology is needed. Generally speaking, convolutional neural networks are often leveraged in these sorts of problems as they allow for a large degree of weight sharing. The photometric light curve is a one dimensional signal of a non-trivial length (i.e. 6000 points). Each convolutional layer will have a number of filters that it convolves with that signal. These filters are usually quite small, 3 points wide in this case. After a certain number of convolutions a pooling layer will be used to reduce the dimensionality of the problem. The final layers will be standard dense layers that then map an observation to class prediction via thresholding activation function such as a sigmoid or softmax (the multi-class variant of the sigmoid). The topology used here is built with five successive modules of two 1D convolution layers with kernel length 3 and 64 filters followed by a max pooling layer with pool size 2. Afterwards the outputs are flattened and fed to 3 fully connected layers of width 64 before the output layer. All non-output layers use the rectified linear unit (ReLU) activation function, whereas the output layer uses the sigmoid activation function. A graphical depiction of this topology can be found in Figure 3.5.

*TODO: Paragraph discussing topology figure interpretation*

Regarding training, network training will continue until 30 epochs pass without improvement to validation performance. The intent is to account for any variance in the amount of training needed autonomously so that many models can be executed simultaneously and without human tuning. The batch size was set to 128 observations per batch in an attempt to find a trade-off between runtime and randomness of training batch contents. The Adam optimizer was used and default learning rate used with it. Dropout was kept in mind as a regularization technique, though would only be employed if the validation performance significantly diverged from the training performance of the model in question. In practice, it was not needed throughout the course of this research. This

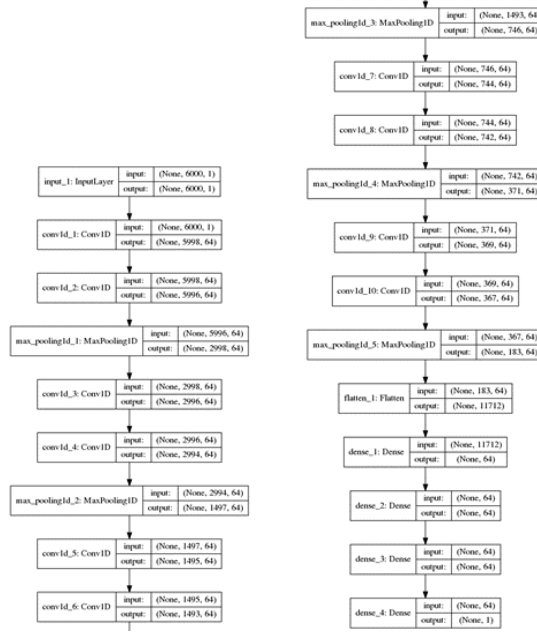


Figure 3.5: Diagram of convolutional neural network topology used to associate photometric observations to RSO

ANN was implemented, trained, and tested using the Keras Python library [12]. The code used can be found in Appendix C.

### 3.4 HPC Resources

The Maui High Performance Computing Center’s (MHPCC’s) computing cluster was used for the experiments performed here. Its own site described it by saying: “Hokulea is an IBM POWER8 server with high-performance, high-bandwidth, NVIDIA GP100 graphical processing units (GPUs). Utilizing the GPUs, Hokulea will provide approximately 690 TFLOPs of supercomputing. As its name implies, it will be used as a test system to evaluate the performance of this novel architecture for DoD-specific software” [2]. The specific performance characteristics for Hokulea can be found in Table 3.2. In this study Hokulea is leveraged to train and evaluate the ANN for multiple replications and folds of the experiments described in the proceeding sections.

Also of interest is a Python framework for experiment execution on the HPC. Cluster experiment was developed at MHPCC and is used here to mitigate the complexity of the cluster’s queuing and management system, known as the Portable Batch System (PBS). The code allows one to provide desired factors and levels of their experiment, where the meaning of those parameters is interpreted by the user’s own code. This provided a great deal of flexibility in development. Cluster experiment also allows the user to set the desired number of replications of each combinations of levels. The cluster experiment code can be found in Appendix D. After execution, the results are concatenated into a single comma-separated file and then processed. For the baseline and maneuvering experiments, a post-processing script was then run on that CSV. Said script can be found in Appendix E. The closely spaced objects experiment produced a small enough results file that it was post-processed by hand.

Table 3.2: Hokulea compute cluster characteristics and performance [2].

| hokulea.mhpcc.hpc.mil      |                                                   |                                     |
|----------------------------|---------------------------------------------------|-------------------------------------|
| IBM POWER8 - 690 TFLOPS    |                                                   |                                     |
|                            | Login Nodes                                       | GPU Accelerated Compute Nodes       |
| Total Cores   Nodes   GPUs | 20   1   0                                        | 640   32   128                      |
| Operating System           | RHEL 7                                            |                                     |
| Cores/Node                 | 20                                                | 20 + 4 GPU<br>(4 x 3.538 GPU cores) |
| Core Type                  | IBM POWER8                                        | IBM POWER8<br>+NVIDIA Tesla P100    |
| Core Speed                 | 2.86 GHz                                          |                                     |
| Memory/Node                | 256 GBytes                                        | 256 GBytes<br>+(4 x 16 GBytes)      |
| Accessible Memory/Node     | 240 GBytes                                        | 244 GBytes<br>+(4 x 16 GBytes)      |
| Memory Model               | Distributed                                       |                                     |
| Interconnect Type          | Mellanox SB7700 36 X EDR 100 Gbits/sec InfiniBand |                                     |
| Workspace                  | 250 TBytes                                        |                                     |

### 3.5 Baseline Association of RSOs

This experiment addresses the first research question: Are there any cases where a DNN can use photometry to correctly associate observations to the corresponding RSO? The machine learning task used to answer that question is pairwise classification of 8 GEO RSOs where those RSOs are all station keeping and spaced apart from one another. These 8 RSOs will have different geometries, materials, and attitude profiles. Table 3.3 contains the

photometric properties of each of the 8 RSOs in this scenario. This problem is perfectly and easily solved via metric data, as even the nearest neighbor method ought to produce correct association in such a benign case. However it is doubtful that photometric observations will be easier to associate in other cases, so this serves as a sort of theoretical maximum performance. A geographic depiction of the catalog used for this experiment is shown in Figure 3.6.

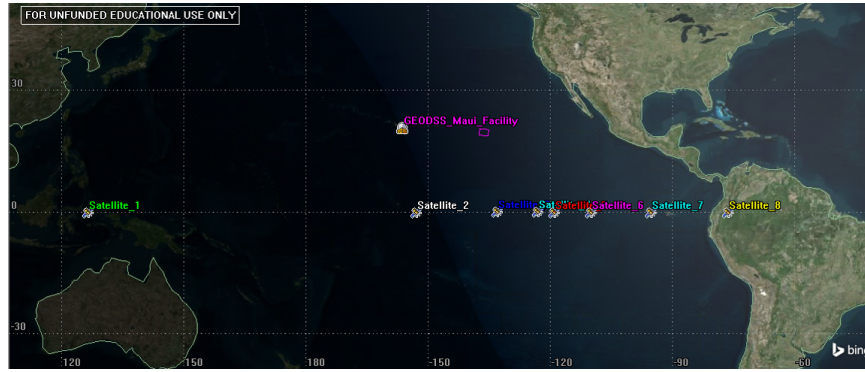


Figure 3.6: Station keeping catalog used in baseline association experiment.

These 8 RSOs are considered pairwise so that the ANN is only ever trained on a two class problem. Note that there are  $\binom{8}{2} = 28$  RSO pairings considered here. The intent is to determine which pairs of RSOs are most difficult to distinguish in an attempt to understand the utility and capability of photometric light curves.

For a specific RSO pair of the 28 pairs under consideration, 3-fold cross validation is used with 25% of the training fold observations used for the validation set. This process will be replicated 10 times with different random seeds so that 30 performance estimates are available.

The code used to create the Astrogator ephemeris, as well as to launch the experiment on the HPC, is given in Appendix F. With a dataset generated and experiment executed, the

Table 3.3: RSO shapes, materials, and attitude profiles used in this experiment.

| RSO Name    | Original Name | Attitude Profile                   | Material     | Geometry   |
|-------------|---------------|------------------------------------|--------------|------------|
| Satellite 1 | JCSAT 16      | Nadir align w/ orbit normal const. | Aluminum MLI | GEOComm    |
| Satellite 2 | INTELSAT 907  | Nadir align w/ orbit normal const. | Aluminum MLI | LEOImaging |
| Satellite 3 | GOES 13       | Nadir align w/ orbit normal const. | Solar Panel  | GEOComm    |
| Satellite 4 | AMC 16        | Nadir align w/ orbit normal const. | Solar Panel  | LEOImaging |
| Satellite 5 | GOES 16       | Nadir align w/ sun const.          | Aluminum MLI | GEOComm    |
| Satellite 6 | DIRECTV 11    | Nadir align w/ sun const.          | Aluminum MLI | LEOImaging |
| Satellite 7 | MEXSAT 3      | Nadir align w/ sun const.          | Solar Panel  | GEOComm    |
| Satellite 8 | GALAXY 15     | Nadir align w/ sun const.          | Solar Panel  | LEOImaging |

results will be aggregated and presented. The intent is to determine in which cases one can reasonably expect RSOs to be distinguishable, at least provided the quantity and quality of data generated here. Armed with this knowledge, this research will then assess the utility of photometric data in distinguishing closely spaced objects.

### 3.6 Association of Closely Spaced RSOs

This experiment addresses the second research question: Can a DNN use photometry to associate observations of RSOs even if those RSOs are closely spaced? The machine learning task used to answer that question is classification of one specific GEO RSO against each of 7 others. The intent is to determine if proximity has an impact on the photometric association process. The RSOs in this scenario will all station keep at increasing distances; the stations assigned to each are shown in Table 3.4. A graphic showing this catalog can be found in Figure 3.7.

Satellite 1 will serve as the baseline objects in this experiment and it will be examined against satellites 2 through 8. This amounts to 7 different two-class classification problems:

Table 3.4: Longitude of assigned station for each RSO in the closely spaced RSOs scenario.

| Satellite Number | 1              | 2               | 3              | 4              | 5              | 6              | 7              | 8              |
|------------------|----------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Longitude        | $-150.0^\circ$ | $-150.01^\circ$ | $-150.1^\circ$ | $-150.2^\circ$ | $-150.3^\circ$ | $-150.4^\circ$ | $-150.5^\circ$ | $-150.6^\circ$ |

Satellite 1 vs Satellite  $i$  for  $i = 2, 3, \dots, 8$ . The photometric qualities of these RSOs will be selected so that satellite 1 should be distinguishable from the remainder of the catalog. The satellites other than satellite 1 will all share the same photometric qualities so that the only factor varying in this experiment is proximity. As for specific values used for those photometric qualities, they will be informed by the results of the baseline experiment.

For each of these individual two-class problems, 6-fold cross validation will be employed and will be replicated 5 times so that there are 30 performance estimates for each problem. These results will then be collected and analyzed.

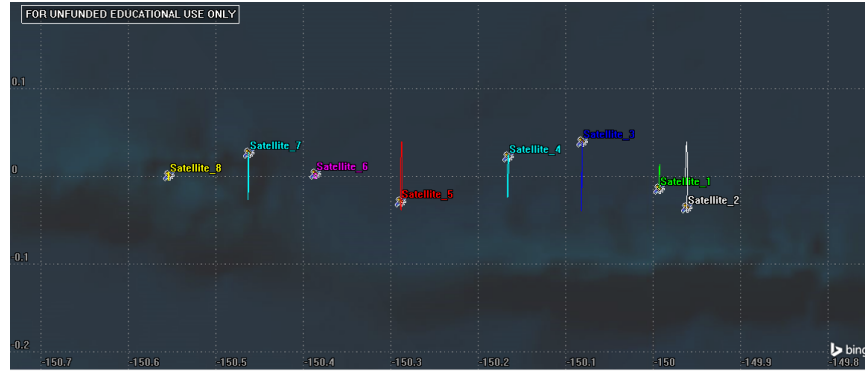


Figure 3.7: Station keeping catalog used in closely spaced objects experiment.

The code used to create the Astrogator ephemeris, as well as to launch the experiment on the HPC, is given in Appendix G. With the results of this experiment in mind, one should be able to quantify the utility of this method in association of observations of closely spaced objects. It is reasonable to anticipate that the photometric signature would remain just as

salient a feature in object differentiation regardless of proximity (supposing it is not closer than the sensor's resolution of course). With that in mind, one would expect a high, though likely not 100%, accuracy between each assessed RSO pair, and that that accuracy does not degrade with proximity. If RSOs could be differentiated in the previous experiment that trend is anticipated to continue here. If RSOs could not be differentiated in the previous experiment it is not anticipated to improve here.

### **3.7 Association of Maneuvering RSOs**

This experiment addresses the third research question: Can a DNN use photometry to associate observations of RSOs even if those RSOs are maneuvering? The machine learning task used to answer that question is pairwise classification of 8 GEO RSOs where those RSOs will periodically permute stations with one another. The generation of the dataset itself will closely follow that used in the baseline described earlier in this chapter. The scenario will be a year long with observations taken every 6 hours (as constrained by lighting of the sensor and RSOs) and the same shapes, materials, and attitude profiles will be used. The difference is that Astrogator is not used to command the RSOs to keep station throughout the whole year. Instead every 31 days the RSOs will change to a station occupied by a different RSO. The schedule is given by the ordered list: JCSAT 16, INTELSAT 907, GOES 13, AMC-16, GOES 16, DIRECTV 11, MEXSAT 3, and GALAXY 15. Each time a given RSO maneuvers it will advance one position in a randomized version of the above list.

The way that Astrogator accomplishes this is not dissimilar to the way it kept station in the baseline scenario. The target sequence used will still amount to a Hohmann transfer, though the tolerance is initially loosened to allow Astrogator to converge for large-scale maneuvers. The reason being is that, due to the scale of these maneuvers, converging on appropriate maneuvers can be difficult. Progressively letting Astrogator step closer to the desired station ensures that tight tolerances are only expected when maneuver magnitude

is also relatively small. This is done in four successive target sequences, with tolerances equal to  $10^\circ$ ,  $5^\circ$ ,  $1^\circ$ , and  $0.05^\circ$  longitude, respectively. A graphic showing the resulting catalog can be seen in Figure 3.8.

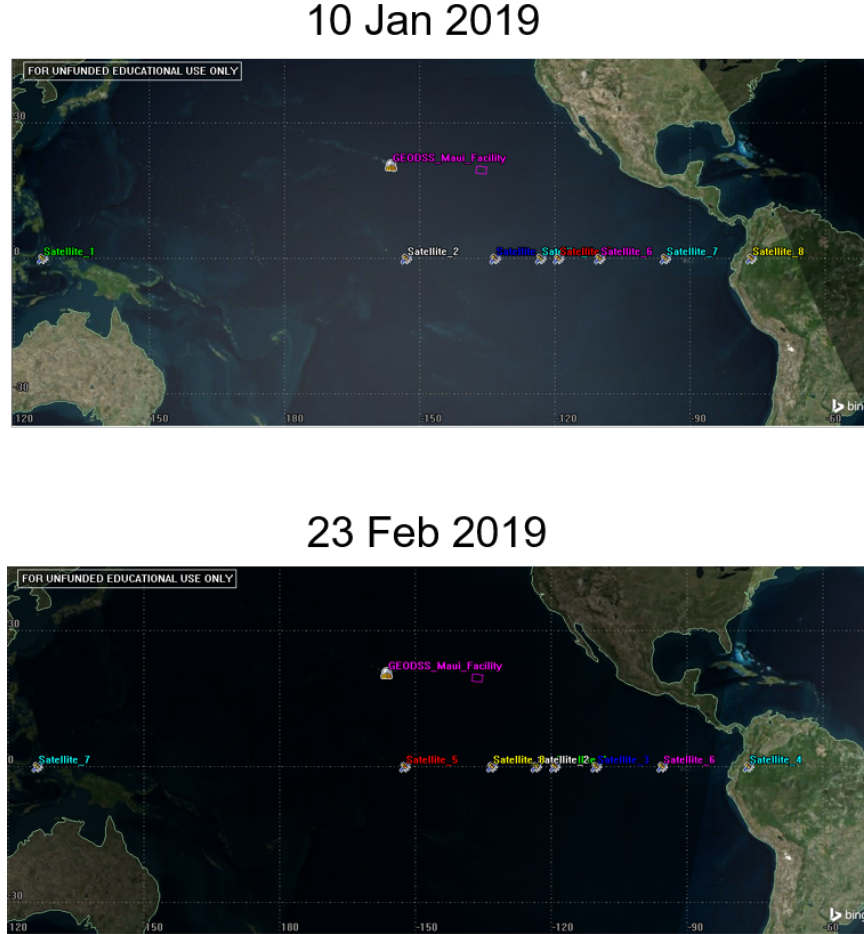


Figure 3.8: Maneuvering catalog where RSOs permute stations every 31 days. Used in the maneuvering experiment.

Similar to the baseline experiment, this experiment considers every pair of RSOs (28 in total). For each pair, 3-fold cross validation is used with 25% of the training folds used for validation. Then 10 replications of the training and evaluation are run to yield 30 performance estimates.

The code used to create the Astrogator ephemeris, as well as to launch the experiment on the HPC, is given in Appendix H. As in the previous experiment, it is not anticipated that RSOs that could not be distinguished in a non-maneuvering catalog will become distinguishable here. For those that could be accurately associated previously, however, it is suspected that maneuvering and the resulting change of attitude alters the RSO's photometric signature significantly enough to cause it to be incorrectly labeled by the trained ANN. In addition to determining when this method is effective it would be interesting to note the expected accuracy degradation due to an RSO maneuvering.

This chapter has covered the dataset generation process, ANN topology, and HPC resources used for the experiments conducted. Afterwards each of the three experiments were discussed in detail. This includes the baseline station-keeping scenario, association of closely spaced objects scenario, and association of maneuvering objects scenario. Altogether, this research will assess ANNs trained on RSO photometric signature in a non-maneuvering baseline, a closely spaced objects scenario, as well as a maneuvering scenario. These experiments collectively serve as a way to measure the effectiveness of photometry in observation association.

## IV. Results

The experiments described in the previous chapter were executed and the results collected. The following sections contain a description of these results, as well as a brief discussion of their implications and outstanding or unexpected findings. As previously, these sections correspond to a baseline station keeping scenario, a closely spaced RSOs scenario, and finally a maneuvering scenario.

### 4.1 Baseline Association of RSOs

Regarding the station keeping scenario, recall that this experiment consisted of a catalog of 8 station keeping GSO RSOs. A simulation year's worth of photometric observation were generated for those RSOs, and a deep convolutional neural network was evaluated on each RSO pairing. The primary metric used in evaluation of these ANN classifiers is accuracy, that is, the percentage of correct labels the classifiers produced on the respective testing folds. Recall from the previous chapter that each RSO pair will have 30 such accuracy values. The mean and standard deviation of those values are presented in Table 4.1. Each cell represents the mean or standard deviation for the classifiers that compared satellite  $i$  to satellite  $j$  where  $i$  and  $j$  are the row and columns numbers of that cell, respectively. This is why the diagonal is excluded (no RSO is ever compared to itself). Similarly, the values below the diagonal are aggregated with those above the diagonal and are thus excluded (the satellite 2 vs satellite 1 cell is combined with the satellite 1 vs satellite 2 cell). The colors correspond to relative goodness. That is, the worst performance values are in red and the best are in green. For the accuracy means, bigger is better. For the accuracy standard deviations, smaller is better.

From these results, a somewhat non-intuitive pattern appears to emerge. Recall that 50% accuracy is the worst one should expect in a 2-class problem, so the cases in Table

Table 4.1: Accuracy means and standard deviations for every RSO pairing in the baseline association experiment. Each cell summarizes a sample of size 30.

| Mean Accuracy |             |             |             |             |             |             |             |             |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|               | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1   | -           | 67.19%      | 82.25%      | 89.45%      | 87.77%      | 89.60%      | 76.94%      | 81.64%      |
| Satellite 2   | -           | -           | 100.00%     | 99.25%      | 99.63%      | 93.95%      | 97.96%      | 64.85%      |
| Satellite 3   | -           | -           | -           | 100.00%     | 54.76%      | 98.61%      | 52.76%      | 99.94%      |
| Satellite 4   | -           | -           | -           | -           | 99.66%      | 54.59%      | 97.24%      | 70.96%      |
| Satellite 5   | -           | -           | -           | -           | -           | 95.85%      | 62.86%      | 98.17%      |
| Satellite 6   | -           | -           | -           | -           | -           | -           | 95.81%      | 77.99%      |
| Satellite 7   | -           | -           | -           | -           | -           | -           | -           | 99.57%      |
| Satellite 8   | -           | -           | -           | -           | -           | -           | -           | -           |

| Standard Deviation of Accuracy |             |             |             |             |             |             |             |             |
|--------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                                | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1                    | -           | 10.85%      | 7.63%       | 11.87%      | 1.92%       | 8.42%       | 2.57%       | 6.17%       |
| Satellite 2                    | -           | -           | 0.00%       | 3.83%       | 0.51%       | 11.13%      | 1.44%       | 8.54%       |
| Satellite 3                    | -           | -           | -           | 0.00%       | 4.00%       | 7.46%       | 4.69%       | 0.08%       |
| Satellite 4                    | -           | -           | -           | -           | 0.21%       | 1.71%       | 10.16%      | 10.60%      |
| Satellite 5                    | -           | -           | -           | -           | -           | 8.62%       | 9.00%       | 0.47%       |
| Satellite 6                    | -           | -           | -           | -           | -           | -           | 10.39%      | 1.96%       |
| Satellite 7                    | -           | -           | -           | -           | -           | -           | -           | 0.87%       |
| Satellite 8                    | -           | -           | -           | -           | -           | -           | -           | -           |

4.1 with means near 50% correspond to classifiers failing to significantly tell the RSOs in question apart. As far as a proof of concept, certainly some RSO pairings appear quite distinguishable (e.g. Satellite 2 vs Satellite 3). That is to say that it would appear there are cases where photometric observations are a reliable way of distinguishing RSOs. It should be noted that any model trained on observations of satellite 1 performs poorly, relatively speaking. The reason is not immediately obvious, though if one recalls Figure 3.6, Satellite 1 is separated from the rest of the catalog by a large distance. This could lead to significantly different viewing aspects, or exposed facets, amounting to a photometric signature indistinguishable from the rest of the catalog. With all that in mind, what causes some cases to work better than others is hopefully made more clear in Figure 4.1.

Figure 4.1 aggregates the samples summarized in Table 4.1 based on the way the RSOs differ. So, for instance, every RSO pair that differ only in material are combined in the first

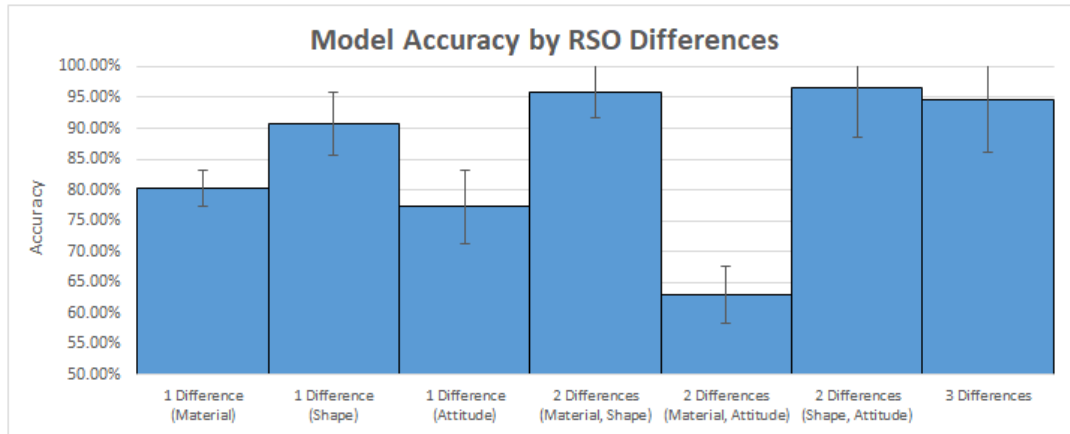


Figure 4.1: Mean accuracy arranged by manner in which each RSO pair differs. Bar height gives mean accuracy given those differences, and error bars show standard deviation. Note that each bar describes 4 RSO pairs, hence a sample of size 120.

bar. This would be satellite 1 vs satellite 3, satellite 2 vs satellite 4, satellite 5 vs satellite 7, and satellite 6 vs satellite 8 (see Table 3.3). Each bar will have 4 RSO pairs included. Since each RSO pair is itself a sample of size 30, each bar described a sample of size 120. The height is the mean accuracy of that sample, and the error bars describe the sample standard deviation.

With that description in mind, the generally decent performance of this technique is made clear. Aside from the 2 differences - material and attitude case, most bars seem accurate. Of the cases that don't perform as well, they all hold RSO shape constant. That is, the performance of this technique appears to degrade in cases where RSOs have the same geometry, even if the material and/or attitude profile varies. This would concur with intuition. Even though it was anticipated that all of these factors would impact an RSO's photometric signature, this gives clear evidence that geometry has the most significant impact.

## 4.2 Association of Closely Spaced RSOs

When considering the subsequent experiment regarding closely spaced RSOs, we start with RSOs that seem easily distinguished. In this case RSOs that differ in both material and shape are taken, as this case appears to perform very well in the station keeping experiment.

The specific RSOs used in the closely spaced RSOs scenario are satellite's 2 and 3. These differ in both geometry and material though have the same attitude profile. The results obtained can be seen in Table 4.2. The table is read in the same manner as Table 4.1, however with only a single row. Since every RSO pairing wasn't considered in this experiment, 6-fold cross validation was used instead of 3-fold so that the resulting accuracy samples were still each of size 30.

Table 4.2: Accuracy means and standard deviations for every RSO pairing in the closely spaced objects association experiment. Each cell summarizes a sample of size 30.

| Mean Accuracy |             |             |             |             |             |             |             |             |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|               | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1   | -           | 87.04%      | 88.43%      | 88.93%      | 88.04%      | 87.90%      | 88.65%      | 87.76%      |

| Standard Deviation of Accuracy |             |             |             |             |             |             |             |             |
|--------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                                | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1                    | -           | 3.10%       | 3.49%       | 2.77%       | 2.47%       | 3.26%       | 2.63%       | 3.15%       |

One notices that these performance estimates are all near that of the previous experiment. This is as predicted, as proximity should not impact the performance of this technique. To further strengthen that claim there does not seem to be a pattern of improvement or degradation as the column's satellite number increases. So even gradually increasing or decreasing RSO distance doesn't appear to impact this technique. At the extreme satellite 1 and 2 are practically station sharing. Indeed both RSO's drift tolerances include the center of each other's station. All of that to say that, provided the RSOs were

distinguishable initially, they still can be distinguished when placed close together. This certainly can't be true within the angular resolution of the sensor. But it seems reasonable to claim that photometric-based observation association would be effective up to that point.

### 4.3 Association of Maneuvering RSOs

For the maneuvering experiment the same reporting mechanisms are used as in the baseline experiment. The means and standard deviations of the resulting accuracy samples can be found in Table 4.3. The table is interpreted in the same manner as table 4.1.

Table 4.3: Accuracy means and standard deviations for every RSO pairing in the maneuvering association experiment. Each cell summarizes a sample of size 30.

| Mean Accuracy |             |             |             |             |             |             |             |             |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|               | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1   | -           | 69.85%      | 52.58%      | 67.46%      | 63.42%      | 73.96%      | 60.26%      | 71.55%      |
| Satellite 2   | -           | -           | 69.02%      | 50.67%      | 58.83%      | 50.83%      | 59.77%      | 57.72%      |
| Satellite 3   | -           | -           | -           | 67.45%      | 58.60%      | 72.42%      | 55.76%      | 72.93%      |
| Satellite 4   | -           | -           | -           | -           | 58.72%      | 51.85%      | 58.64%      | 58.70%      |
| Satellite 5   | -           | -           | -           | -           | -           | 59.38%      | 50.19%      | 68.74%      |
| Satellite 6   | -           | -           | -           | -           | -           | -           | 62.86%      | 53.91%      |
| Satellite 7   | -           | -           | -           | -           | -           | -           | -           | 67.22%      |
| Satellite 8   | -           | -           | -           | -           | -           | -           | -           | -           |

| Standard Deviation of Accuracy |             |             |             |             |             |             |             |             |
|--------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                                | Satellite 1 | Satellite 2 | Satellite 3 | Satellite 4 | Satellite 5 | Satellite 6 | Satellite 7 | Satellite 8 |
| Satellite 1                    | -           | 2.79%       | 2.04%       | 2.65%       | 3.17%       | 3.04%       | 2.87%       | 2.77%       |
| Satellite 2                    | -           | -           | 4.06%       | 1.04%       | 4.07%       | 1.11%       | 4.62%       | 3.48%       |
| Satellite 3                    | -           | -           | -           | 5.26%       | 4.67%       | 4.75%       | 3.22%       | 3.73%       |
| Satellite 4                    | -           | -           | -           | -           | 7.08%       | 2.35%       | 4.22%       | 5.87%       |
| Satellite 5                    | -           | -           | -           | -           | -           | 3.64%       | 1.09%       | 5.64%       |
| Satellite 6                    | -           | -           | -           | -           | -           | -           | 3.36%       | 3.64%       |
| Satellite 7                    | -           | -           | -           | -           | -           | -           | -           | 3.83%       |
| Satellite 8                    | -           | -           | -           | -           | -           | -           | -           | -           |

At a glance, one notices the performance degradation due to the RSO maneuvers. Though some pairings remain somewhat distinguishable, many pairing give accuracies near 50%. Again, this means that those RSO pairs could not be distinguished any more accurately than a random guess. That maneuvering RSOs are more difficult to distinguish

is not surprising, though perhaps the degree to which those maneuvers impact an RSOs photometric signature is. Figure 4.2 was constructed for the same purpose as in the station keeping experiment: to determine in what cases specifically will photometric-based observation association be effective. Unfortunately, the differences between bars are mostly within the expected variances of each case. As far as viability, it certainly appears that some cases perform better than a 50% random classifier. Though this could be noise, with 120 samples it is reasonable to conjecture that this is evidence of a learned relationship between the photometric signature and RSO identity. It is possible that more observations are simply needed from different perspectives in order to prepare the classifier for the dynamic behavior presented by maneuvering RSOs. If that were true, then one would expect performance to improve significantly as more training data was used.

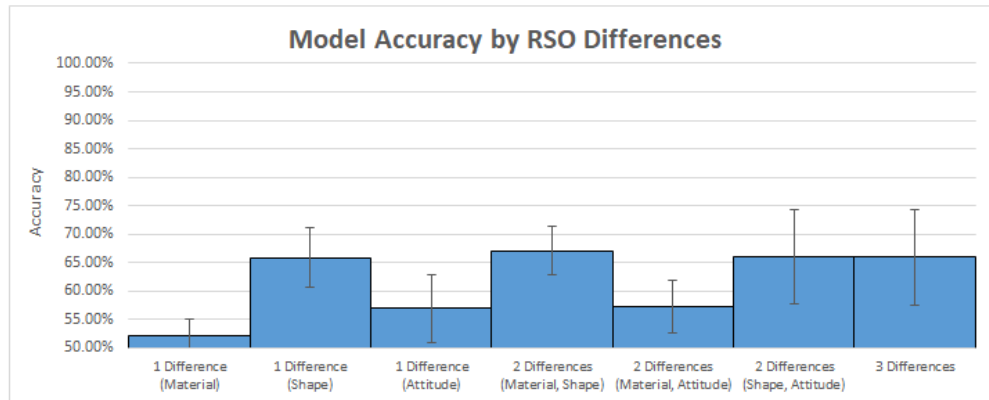


Figure 4.2: Mean accuracy by RSO pairing differences. Bar height gives mean accuracy given those differences, and error bars show standard deviation. Note that each bar describes 4 RSO pairs, hence a sample of size 120.

In total, these results demonstrate the viability of photometric-based observation association in some cases, specifically that of RSOs with fundamentally different

geometries. Furthermore this appears to be agnostic of RSO spacing so that, in those cases with different RSO geometries, this technique could provide the community with valuable capability, as association of closely spaced objects is a known limitation of traditional astrometric-based techniques. Finally, for maneuvering RSOs the utility of photometry appears significantly degraded. That being said, the technique yielded sufficient performance to warrant further research. It is possible that further training in more diverse scenarios could produce a classifier with similar performance to that seen in the station keeping scenario.

## **V. Conclusions and Recommendations for Future Work**

This document has covered the research questions and hypothesis under consideration, a brief treatment of the background of both SSA and machine learning, the methodology used for the three experiments conducted, as well as the results of those experiments. With this all in mind, the question once again is posed: Are there any cases where photometry is a viable method to distinguish RSOs? And if so, what are the limits of this technique and how practical are those limit?

### **5.1 Conclusions**

First, there are indeed cases where photometry can be reliably used to distinguish GEO RSOs. In particular, it was noted in the previous chapter that RSOs with different geometries were most likely to be distinguishable. In a limited sense those that differed by material and/or attitude profile could also be distinguished, but geometry was the most directly correlated to performance. The operational implications of this conclusion are limited, as non-maneuvering catalogs without closely spaced objects are easily addressed via existing astrometric techniques. Still, this demonstrates the validity of photometric based association and leads one to consider its application to less trivial cases.

The first of those less trivial cases is one containing closely spaced RSOs. As was described previously, this is a known limitation of current association techniques. Here it was demonstrated that, provided the RSOs were originally distinguishable, spacing doesn't degrade the accuracy of a photometric based associator. That is, for GEO RSOs with different geometries, photometry could be used to distinguish them even if they were stationed very close (less than  $0.05^\circ$  longitude) together. An important caveat is that the associator must be trained on the RSO's photometric signature beforehand. The

generalization of that signature has not yet been fully studied, meaning performance could degrade significantly if the training set differs contextually from that of the evaluation data.

With that in mind, the final case considered is that of maneuvering RSOs. Regarding the generalization of an RSO's photometric signature, this is an extreme form of that generalization. The performance demonstrated here was not conclusive and implies that either the signature does not generalize at all, or else insufficient data was captured to generalize to the contexts used in evaluation. Given that better than random performance was observed in some cases, the latter is most likely the case. It seems reasonable that, provided sufficient photometric data from different possible maneuvers, lighting conditions, observation aspects, etc., one could expect a greater degree of generalization in that photometric associator and thus higher testing accuracy.

## **5.2 Recommendations**

This naturally implies several major avenues for future research. The generalization of an RSO's photometric signature is the most immediate such avenue. First, the ability to generalize in time, even in a non-maneuvering scenario, needs to be demonstrated before this technique is viable in any operational context. Additionally, generalization of the photometric signature to different maneuvers is a more difficult, though very interesting problem. Likely many potential "universes" should be created, each constituting a possible maneuver of that RSO. Aggregating datasets in this manner would hopefully eventually lead a deep neural network to learn a more general mapping from photometric signature to RSO identity, one that would be effective over many different maneuvers.

It may be that photometric signatures, as framed here, are simply insufficient to accomplish the tasks described. A natural follow-on would be to consider multi-channel signatures. The most basic form of this would be to divide the visible spectrum into red, blue, and green light components. More extreme still would be to create a large number of channels with a very fine spectral resolution, potentially even expanding beyond the

visible spectrum. With such a sensor in mind, then, does material become a more salient feature? And if so how much capability can be added to the above approaches? This may be necessary to achieve the previous paragraph's stated goal of photometric signature generalization.

Finally, rather than consider just a single ANN topology, many different topologies ought to be evaluated. It is possible that the capability gaps discovered here are not limitations of photometry itself, but merely of the machine learning method employed. If that is the case, more suited topologies should be considered along with the larger and more representative datasets suggested previously. Altogether the intent being to create and demonstrate a technique for observation association from RSO photometric signature that is as general as possible so as to be as useful as possible to the SSA community.

## Appendix A: Load Ephemeris

Listing A.1: Python Script to Populate STK Scenario with Astrogator Ephemeris

```
1 # Title: Load Ephemeris to Scenario
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Helper script to populate an STK scenario from a set of generated ephemeris files. That scenario will
6 # require user attention before being used to generate a dataset of RSO observations.
7
8 from win32com.client import GetActiveObject
9 from comtypes.client import CreateObject
10 from comtypes.gen import STKUtil
11 from comtypes.gen import STKObjects
12 from comtypes.gen import AgStkGatorLib # Must first be generated
13 from sklearn.preprocessing import LabelEncoder
14 from keras.utils import np_utils
15 import win32com
16 import os
17 import numpy as np
18 import time
19 import multiprocessing
20 import sys
21 import shutil
22
23 # Only run this line once, possibly even just from a command line. This generates the astrogator module
24 # comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18F4017C}"), 1, 0))
25
26 stkLocation = os.path.join("C:/", "Users", "ianmc", "Documents", "STK 11 (x64)")
27
28
29 def startSTK(isVisible=True):
30     uiApplication = win32com.client.Dispatch('STK11.Application')
31     uiApplication.Visible = isVisible
32
33     # Start new instance of STK
34     uiApplication = CreateObject('STK11.Application')
35     uiApplication.Visible = isVisible
36
37     # Get our IAgStkObjectRoot interface
38     root = uiApplication.Personality2
39
40     # Set some of the units to be a little nicer. I prefer seconds.
41     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
42
43     return root, uiApplication
44
45
46 def createTemplateScenario(root):
47     # IAgStkObjectRoot root: STK Object Model Root
48     root.NewScenario('Script.Scenario.Template')
49
50     # Switch units back temporarily
51     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('UTCG')
52
53     startDatetime = '1 Jan 2019 12:00:00.000'
54     stopDatetime = '1 Jan 2020 12:00:00.000'
55     scen = root.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
56     scen.SetTimePeriod(startDatetime, stopDatetime)
57
58     # And reset the units before we return
59     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
60
61
62 def saveTemplateScenario(root, scenarioName):
63     # Make sure a directory is devoted to this scenario
64     saveDir = os.path.join(stkLocation, scenarioName)
65     if os.path.isdir(saveDir):
66         shutil.rmtree(saveDir)
67     os.mkdir(saveDir)
68
69     root.SaveScenarioAs(os.path.join(saveDir, scenarioName))
70
71
72 def loadEphemeris(stkRoot, ephemerisPath):
73     for file in os.listdir(ephemerisPath):
74         if file.endswith(".broken"):
75             print("Can't load broken ephemeris. Skipping...")
76         elif file.endswith(".e"):
77             print("Loading file: " + str(file))
78             satellite = stkRoot.CurrentScenario.Children.New(18, file.split(".")[0])
79             satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
80             satellite2.SetPropagatorType(6) # set to STK external (i.e. ephemeris file)
81
82             p = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorStkExternal)
83             p.Filename = os.path.join(ephemerisPath, file)
84
85             try:
86                 p.Propagate()
87             except Exception:
88                 satellite.Unload()
89                 os.rename(os.path.join(ephemerisPath, file), os.path.join(ephemerisPath,
90                                     file.split(".")[0] + ".broken"))
91                 print("Error on propagate, skipping...")
92
93     # Lighting constraint
94     accessConstraints = satellite.AccessConstraints
```

```

94         light = accessConstraints.AddConstraint(25) # eCstrLighting
95         light.Condition = 0 # direct sun
96
97         # Set attitude
98
99         # Set shape/material
100     else:
101         raise Exception("Error. Unexpected file type: " + str(file))
102
103
104 def addFacility(name, lat, long):
105     facility = stkRoot.CurrentScenario.Children.New(8, name + '.Facility')
106     facility2 = facility.QueryInterface(STKObjects.IAgFacility)
107     pos = facility2.Position
108     pos.AssignGeodetic(lat, long, 0.0) # Latitude, Longitude, Altitude
109     facility2.UseTerrain = True
110     facility2.HeightAboveGround = .05 # km
111     return facility
112
113
114 def addSensor(name, facility, bandLow=0.4, bandHigh=0.9):
115     # Sensor
116     sensor = facility.Children.New(20, name + '.Sensor')
117     sensor2 = sensor.QueryInterface(STKObjects.IAgSensor)
118     commonTasks = sensor2.CommonTasks
119     commonTasks2 = commonTasks.QueryInterface(STKObjects.IAgSnCommonTasks)
120
121     # Can only see if it is dark
122     accessConstraints = sensor.AccessConstraints
123     light = accessConstraints.AddConstraint(25) # eCstrLighting
124     light.Condition = 3 # penumbra or umbra
125
126     # 0 - geometric input
127     # 1 - radiometric input
128     # 2 - sensor output
129     # 3 - sensor off
130     processingLevel = 0
131     losJitterMrad = 1.0
132
133     commonTasks2.SetPatternEOIR(losJitterMrad, processingLevel)
134     commonTasks2.SetPointingFixedAzEl(90, 60, 1)
135
136     # Now for all of the EOIR parameters
137     pattern = sensor2.Pattern.QueryInterface(STKObjects.IAgSnEOIRPattern)
138     bands = pattern.Bands
139     band = bands(0)
140
141     band.BandName = "MainBand"
142     # band.BitDepth = 8 # Not needed
143     # band.CalculatingParameters = False
144     band.DarkCurrent = 6.0
145     # band.DetectorFillFactor = 1.0
146     band.DetectorFullWellCapacity = 160000.0
147     # band.DynamicRange = 3000.000244140625 # Computed/not needed
148     band.DynamicRangeMode = True
149     # band.EffFocall = 11.0 - Computed
150     band.OpticalInputMode = 1 # Input as aperture diameter/f number
151     band.EntrancePDia = 101.0
152     band.Fnumber = 2.15
153     band.HighBandEdgeWL = bandHigh
154     band.HorizontalHalfAngle = 1.23
155     # band.HorizontalIFOV = 15.624682120200712
156     band.HorizontalPixels = 32.0
157     # band.HorizontalPP = 1718.749999998338
158     # band.ImageQuality = 0
159     band.IntegrationTime = 0.01
160     # band.LongDFocus = 0.0
161     band.LowBandEdgeWL = bandLow
162     # band.NEI = 1.0000000036274937e-15 # Computed
163     # band.NumIntervals = 6 - Simulation thing, don't care that much
164     # band.OpticalQualityDataFile = 'C:\\Program Files\\AGI\\STK 11\\EOIR_Databases\\PropertyFiles\\Shape_Files\\2
165     # dGaussian4StdDev64x64.csv'
166     # band.OpticalQualityDataFileFrequencySampling = 5.0
167     # band.OpticalQualityDataFileSpatialSampling = 5.0
168     # band.OpticalTransmission = 1.0
169     # band.OpticalTransmissionMode = 0
170     # band.OpticalTransmissionSpectralResponseFile = 'C:\\Program Files\\AGI\\STK 11\\EOIR_Databases\\PropertyFiles\\
171     # Optical_Transmission_Files\\ARGlass_Trans.srf'
172     # band.QEFile = 'C:\\Program Files\\AGI\\STK 11\\EOIR_Databases\\PropertyFiles\\QE_Files\\BackIllumSi_QE.srf'
173     band.QEMode = 0 # Use Band effective mode
174     band.QEValue = 0.3
175     # band.QSS = 100.0 # Not needed
176     band.QuantizationMode = 0 # Full-well + noise mode
177     band.RadParamLevel = 1 # Use low-level mode
178     band.ReadNoise = 12.0
179     # band.RenderBand = True
180     # band.RMSWavefrontError = 0.0
181     # band.RSRUnits = 1
182     band.SaturationMode = 0 # Irradiance mode is best for non-resolved images
183     # band.Saturations = 0 # NOTE: This is an object collection
184     # band.SEI = 3.000000204853026e-12
185     # band.Sensitivities = 0 # NOTE: This is an object collection
186     band.SimulateQuantization = True
187     band.SpatialInputMode = 0 # FOV + number of pixels
188     # band.SpectralShape = 0
189     # band.SystemRelativeSpectralResponseFile = 'C:\\Program Files\\AGI\\STK 11\\EOIR_Databases\\PropertyFiles\\
190     # RSR_Files\\Pan_RSR.srf'
191     band.VerticalHalfAngle = 1.61
192
193     band.VerticalPixels = 32.0
194     # band.VerticalPP = 1718.749999998338
195     # band.VerticalIFOV = 15.624682120200712
196
197     # band.Wavelength = 0.55
198     # band.WavelengthType = 1
199
200     return sensor

```

```

199
200 if __name__ == "__main__":
201     print("Starting Script...")
202
203     # tleFile = os.path.abspath('test_catalog.tle')
204     tleFile = os.path.abspath('geo_catalog_31july2017.tle')
205     if len(sys.argv) == 2:
206         ephemerisPath = os.path.abspath(sys.argv[1])
207         stkGUIVisible = False
208
209         stkRoot, uiApp = startSTK(stkGUIVisible)
210         createTemplateScenario(stkRoot)
211         loadEphemeris(stkRoot, ephemerisPath)
212         scen = stkRoot.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
213
214         facility = addFacility('GEODSS_Maui', 20.71, -156.26)
215         sensor = addSensor('Sensor', facility, 0.4, 0.9) # Maui, Hawaii
216
217         # Do things by hand
218         # Add targets to EOIR
219         # Set sensor to penumbra/umbra (script doesn't work)
220         # saveTemplateScenario(stkRoot, 'Script_Scenario_Template_StationKeep')
221         # saveTemplateScenario(stkRoot, 'Script_Scenario_Template_CS0')
222         # saveTemplateScenario(stkRoot, 'Script_Scenario_Template_Maneuver')
223     else:
224         print("Error: Please provide a path to the ephemeris folder.")
225
226     print("Script Complete. Exiting...")

```

## Appendix B: Generate Data

Listing B.1: Python Script to Automate STK Generation of RSO Observations

```
1 # Title: Generate Data
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: This script generate a set of RSO observations from a given STK scenario. This can take quite some time
6 # to run and uses multiprocessing to accomplish the task a bit faster. Most of the runtime is consumed generating the
7 # light curves.
8
9 from win32com.client import GetActiveObject
10 from comtypes.client import CreateObject
11 from comtypes.gen import STKUtil
12 from comtypes.gen import STKObjects
13 from comtypes.gen import AgStkGatorLib # Must first be generated
14 from sklearn.preprocessing import LabelEncoder
15 from keras.utils import np_utils
16 import win32com
17 import os
18 import numpy as np
19 import time
20 import multiprocessing
21
22 # Only run this line once, possibly even just from a command line. This generates the astrogator module
23 #comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18F4017C}"), 1, 0))
24
25 stkLocation = os.path.join("C:/", "Users", "ianmc", "Documents", "STK 11 (x64)")
26
27 def startSTK(isVisible=True):
28     uiApplication = win32com.client.Dispatch('STK11.Application')
29     uiApplication.Visible = isVisible
30
31     # Start new instance of STK
32     uiApplication = CreateObject('STK11.Application')
33     uiApplication.Visible = isVisible
34
35     # Get our IAgStkObjectRoot interface
36     root = uiApplication.Personality2
37
38     # Set some of the units to be a little nicer. I prefer seconds.
39     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
40
41     return root, uiApplication
42
43 def loadTemplateScenario(root, scenarioName="Script-Scenario.Template.sc"):
44     # IAgStkObjectRoot root: STK Object Model Root
45     root.LoadScenario(os.path.join(stkLocation, scenarioName))
46     objCollection = root.CurrentScenario.Children.QueryInterface(STKObjects.IAgStkObjectCollection)
47
48     satList = []
49     nameList = []
50     sensorList = []
51     for i in range(objCollection.Count):
52         temp = objCollection(i).QueryInterface(STKObjects.IAgStkObject)
53         if temp.ClassName == 'Facility':
54             for i in range(temp.Children.Count):
55                 sensorList.append(temp.Children(i))
56                 # print("Loading facility:" + str(temp.InstanceName) + "...")
57             elif temp.ClassName == 'Satellite':
58                 # print("Loading satellite " + str(temp.InstanceName) + "...")
59                 satList.append(temp)
60                 nameList.append(temp.InstanceName)
61
62     return satList, nameList, sensorList
63
64 def getMetricData(sat, streakStart, streakEnd):
65     sphFeatures = ['Time', 'Spherical.RightAscension', 'Spherical.Declination']
66     geodeticFeatures = ['Lat', 'Lon']
67     sunFeatures = ['Azimuth', 'Elevation']
68     pointingFeatures = ['Parent Rel Azimuth', 'Parent Rel Elevation']
69
70     noiseStdev = 2.0 / (60.0 * 60.0) # 2 arc-seconds noise stdev
71
72     # Take a sample at the beginning and end of the streak
73     rtnArray = np.zeros(shape=(1, 2, len(sphFeatures + geodeticFeatures + pointingFeatures) + 1))
74
75     # Pull spherical coordinates
76     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
77     info = dp.Item('Parameter Set: Orbit').QueryInterface(STKObjects.IAgDataProviderGroup)
78     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
79     prov = group.Item('Orbit').QueryInterface(STKObjects.IAgDataPrvTimeVar)
80     sphResults = prov.ExecSingleElementsArray([streakStart, streakEnd], sphFeatures)
81     for i in range(len(sphFeatures)):
82         rtnArray[0, :, i] = sphResults.getArray(i)
83
84     # Don't noise up the time please
85     if i != 0:
86         rtnArray[0, :, i] += np.random.normal(0, noiseStdev, len(rtnArray[0, :, i]))
87
88     # Pull geodetic coordinates
89     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
90     info = dp.Item('LLA State').QueryInterface(STKObjects.IAgDataProviderGroup)
91     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
```

```

95 prov = group.Item('Fixed').QueryInterface(STKObjects.IAgDataPrvTimeVar)
96 geodeticResults = prov.ExecSingleElementsArray([streakStart, streakEnd], geodeticFeatures)
97 for i in range(len(geodeticFeatures)):
98     rtnArray[0, :, i + len(sphFeatures)] = geodeticResults.getArray(i)
99     rtnArray[0, :, i] += np.random.normal(0, noiseStdev, len(rtnArray[0, :, i]))
100
101 # Pull sensor pointing angle
102 facility = sat.Parent.Children[0]
103 sensor = facility.Children[0]
104 dp = sensor.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
105 prov = dp.Item('Boresight AzEl').QueryInterface(STKObjects.IAgDataPrvTimeVar)
106 pointingResults = prov.ExecSingleElementsArray([streakStart, streakEnd], pointingFeatures)
107 for i in range(len(pointingFeatures)):
108     rtnArray[0, :, i + len(sphFeatures) + len(geodeticFeatures)] = pointingResults.getArray(i)
109     rtnArray[0, :, i] += np.random.normal(0, noiseStdev, len(rtnArray[0, :, i]))
110
111 # Pull sun az and el
112 dp = facility.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
113 prov = dp.Item('Lighting AER').QueryInterface(STKObjects.IAgDataPrvTimeVar)
114 sunResults = prov.ExecSingleElementsArray([streakStart, streakEnd], sunFeatures)
115
116 # Compute phase angle - no noise
117 sunAz = np.array(sunResults.getArray(0))
118 sunEl = np.array(sunResults.getArray(1))
119 satAz = np.array(pointingResults.getArray(0))
120 satEl = np.array(pointingResults.getArray(1))
121
122 # Triangle is approximately isosceles, so this is approximately the phase angle
123 phaseAngle = (((sunAz - satAz) ** 2) + ((sunEl - satEl) ** 2))
124 rtnArray[0, :, -1] = phaseAngle
125
126 return rtnArray
127
128
129 def addSOITask(taskList, streakStart, streakEnd, streakStep):
130     taskList.append([streakStart, streakEnd - streakStep, streakStep, len(taskList)])
131     return taskList
132
133
134 def executeSOITaskList(sat, taskList, scenName, rtnArr, numWorkers):
135     count = 0
136     workerList = [[] for i in range(numWorkers)]
137     for task in taskList:
138         workerList[count % numWorkers].append(task)
139         count += 1
140
141 # Push these jobs off to multiprocessing
142 jobs = []
143 for i in range(numWorkers):
144     if workerList[i] != []:
145         p = multiprocessing.Process(target=getSOIDataWorker, args=(sat.InstanceName, workerList[i], i, scenName))
146         jobs.append(p)
147         p.start()
148
149 # Wait for them all to finish
150 for j in jobs:
151     j.join()
152
153 # Do magic to obtain output
154 for i in range(len(taskList)):
155     xFilename = './Script.Data/x_' + str(i) + '.npy'
156     npArr = np.load(xFilename)
157     rtnArr = np.concatenate([rtnArr, npArr], axis=0)
158
159 # If everything worked then you can clear the temp files
160 for i in range(len(taskList)):
161     xFilename = './Script.Data/x_' + str(i) + '.npy'
162     os.remove(xFilename)
163
164 return rtnArr
165
166
167 def getSOIDataWorker(satName, taskList, workerNum, scenName):
168     try:
169         print("Worker " + str(workerNum) + " starting...")
170
171         photoFeatures = ['Time', 'Effective target irradiance']
172         sunFeatures = ['Azimuth', 'Elevation']
173         pointingFeatures = ['Parent Rel Azimuth', 'Parent Rel Elevation']
174
175         # Start new STK instance
176         workerRoot, workerUI = startSTK(False)
177         satList, nameList, sensorList = loadTemplateScenario(workerRoot, scenName)
178
179         # Aim each sensor at the target
180         isPointed = False
181
182         for sat in satList:
183             if sat.InstanceName == satName:
184                 for sensor in sensorList:
185                     pointSensorAtSat(sensor, sat)
186                     isPointed = True
187             else:
188                 # We don't need any other sat cluttering this scenario
189                 sat.Unload()
190
191         # Error check. Don't produce bad data because pointing failed.
192         if not isPointed:
193             print("Error. Failed to point at target. Aborting...")
194             return
195
196         # Photometric providers
197         dp = sensorList[0].DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
198         prov_1 = dp.Item('EOIR Sensor To Target Metrics').QueryInterface(STKObjects.IAgDataProvider)
199         photoProv_1 = prov_1.QueryInterface(STKObjects.IAgDataPrvTimeVar)
200
201         # Photometric provider
202         facility = sensorList[0].Parent

```

```

203 dp = facility.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
204 tempProv = dp.Item('Lighting AER').QueryInterface(STKObjects.IAgDataProvider)
205 sunProv = tempProv.QueryInterface(STKObjects.IAgDataPrvTimeVar)
206
207 # Photometric provider
208 dp = sensorList[0].DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
209 tempProv = dp.Item('Boresight AzEl').QueryInterface(STKObjects.IAgDataProvider)
210 pointingProv = tempProv.QueryInterface(STKObjects.IAgDataPrvTimeVar)
211
212 count = 0
213 numSkipped = 0
214 jobStartTime = time.time() # Start the time now so you don't count init stuff
215 for task in taskList:
216     streakStart = task[0]
217     streakEnd = task[1]
218     streakStep = task[2]
219     taskNum = task[3]
220
221     # Don't repeat work pls
222     if os.path.exists('./ScriptData/x_' + str(taskNum) + '.npy'):
223         numSkipped += 1
224         continue
225
226     prov.l.PreData = 'Satellite/' + satName + ' Config1'
227     photoResults_l = photoProv.l.ExecElements(streakStart, streakEnd, streakStep, photoFeatures).DataSets
228
229     sunResults = sunProv.ExecElements(streakStart, streakEnd, streakStep, sunFeatures).DataSets
230     pointingResults = pointingProv.ExecElements(streakStart, streakEnd, streakStep, pointingFeatures).DataSets
231
232     length = len(photoResults_l(0).GetValues())
233     rtnArr = np.zeros(shape=(1, length, 3))
234
235     rtnArr[0, :, 0] = photoResults_l(0).GetValues() # Time
236     rtnArr[0, :, 1] = photoResults_l(1).GetValues() # Sensor 1 irradiance
237
238     # Compute phase angle
239     sunAz = np.array(sunResults(0).GetValues())
240     sunEl = np.array(sunResults(1).GetValues())
241     satAz = np.array(pointingResults(0).GetValues())
242     satEl = np.array(pointingResults(1).GetValues())
243
244     # Triangle is approximately isosceles, so this is approximately the phase angle
245     phaseAngle = (((sunAz - satAz) ** 2) + ((sunEl - satEl) ** 2))
246     rtnArr[0, :, -1] = phaseAngle
247
248     # Save things
249     np.save('./ScriptData/x_' + str(taskNum) + '.npy', rtnArr)
250
251     count += 1
252     timeElapsed = time.time() - jobStartTime
253     rate = count / timeElapsed
254     print("Worker " + str(workerNum) + ": " + str(count + numSkipped) + " of " + str(
255         len(taskList)) + " complete.")
256     print("Worker " + str(workerNum) + ": " + str(100 * (count + numSkipped) / len(taskList)) + "% complete.")
257     print("Worker " + str(workerNum) + ": " + str(rate * 60) + " tasks per minute")
258     print("Worker " + str(workerNum) + ": " + str(rate * 60) + " tasks per minute")
259     print("Worker " + str(workerNum) + ": " + str(rate * 60) + " tasks per minute")
260
261 finally:
262     # Done, so return and kill this STK instance
263     workerRoot.CloseScenario
264     del workerRoot
265     del workerUI
266     print("Worker " + str(workerNum) + " complete. Exiting...")
267
268 def getLabels(satName, numSats, le):
269     rtnArray = np.zeros(shape=(1, numSats))
270     rtnArray[0, :] = np_utils.to_categorical(le.transform([satName]), numSats)
271     return rtnArray
272
273 def generateData(stopTime, satList, nameList, sensorList, scenName, bigNameList):
274     # First we have to figure out when we need observations
275     obsTime = 6 # Each observed streak will be 6 seconds long
276     numMetricFramesPerObs = 2
277     numSOIFramesPerObs = 6000
278     timeBetweenObs = 6 * 60 * 60 # 6 hours
279
280     # Make a label encoder (note, this needs every name so we don't collide between scenarios)
281     le = LabelEncoder()
282     le.fit(bigNameList)
283
284     satNum = -1
285     numSkipped = 0
286     loopStartTime = time.time()
287     timeWastedTotal = 0
288     for sat in satList:
289         timeWastedStart = time.time()
290         isInFOV = False
291         soiTasks = []
292         satNum += 1
293         print("Starting satellite " + str(nameList[satNum]) + "...")
294
295         # Check if this one has been completed before
296         if os.path.exists('./Datagen.Results/' + nameList[satNum] + "_soi.npy") or os.path.exists('./Datagen.Results/
297             Z_NOT_IN_FOV_' + nameList[satNum] + ".npy"):
298             print("Satellite " + str(nameList[satNum]) + " was completed previously. Skipping...")
299             numSkipped += 1
300             continue
301
302         # Helpful progress info
303         if satNum != numSkipped:
304             timeElapsed = time.time() - loopStartTime
305             rate = (satNum - numSkipped) / (timeElapsed - timeWastedTotal)
306             print("Completed Satellite " + str(satNum) + " of " + str(len(satList)))
307             print("Time Elapsed: " + printTime(timeElapsed))
308             print("Current Rate: " + printTime(1 / rate) + " per task")

```

```

308     print("Est Time to Go: " + printTime((len(satList) - satNum) / rate))
309     print("-----")
310     print()
311
312     # Need to store the data as we generate it
313     xMetric = np.zeros(shape=(0, numMetricFramesPerObs, 8))
314     y = np.zeros(shape=(0, len(satList)))
315
316     pointSensorAtSat(sensorList[0], sat)
317     access = computeAccess(sat, sensorList[0])
318     accessDelta = 0.2 # For some reason, the front bit of the access has no SOI defined
319     t = 4 * 24 * 60 * 60 # Start at day 5/ skip 4 days
320     while (t + obsTime) <= stopTime:
321         # First jump to the appropriate access interval
322         i = 0
323         while i < len(access) and t > access[i][1]:
324             i += 1
325
326         if i >= len(access):
327             # Done with this satellite
328             break
329
330         # Next attempt to construct observations within this access interval
331         t = max(t, access[i][0] + accessDelta)
332         if (t + obsTime) <= access[i][1]:
333             # We have a streak's times, so compute the data gathered from it
334             isInFOV = True
335             metricObs = getMetricData(sat, t, t + obsTime)
336             soiTasks = addSOITask(soiTasks, t, t + obsTime, obsTime / numSOIFramesPerObs)
337             labelObs = getLabels(nameList[satNum], len(satList), le)
338
339             xMetric = np.concatenate([xMetric, metricObs], axis=0)
340             y = np.concatenate([y, labelObs], axis=0)
341
342             t += obsTime + timeBetweenObs # Schedule our next observation
343         else:
344             # We need to wait for the next access period
345             if (i + 1) < len(access):
346                 t = access[i + 1][0] + accessDelta # Go immediately in the next access period
347             else:
348                 # End of the timeline for this satellite
349                 break
350
351     # Main runtime consumption - run the SOI/photometry/light curves
352     xSOI = np.zeros(shape=(0, numSOIFramesPerObs, 3))
353     xSOI = executeSOITaskList(sat, soiTasks, scenName, xSOI, numWorkers=6)
354
355     # Save all of this satellite's data to file
356     if isInFOV:
357         np.save("./Datagen-Results/" + nameList[satNum] + ".metric.npy", xMetric)
358         np.save("./Datagen-Results/" + nameList[satNum] + ".label.npy", y)
359         np.save("./Datagen-Results/" + nameList[satNum] + ".soi.npy", xSOI)
360     else:
361         print("Satellite " + str(nameList[satNum]) + " not in FOV. Skipping...")
362         numSkipped += 1
363         timeWastedTotal += time.time() - timeWastedStart
364         np.save("./Datagen-Results/Z.NOT-IN-FOV" + nameList[satNum] + ".npy", [])
365
366 def printTime(seconds):
367     m, s = divmod(seconds, 60)
368     h, m = divmod(m, 60)
369     return "%d Hours %02d Minutes %02d Seconds" % (h, m, s)
370
371
372 def loadEphemeris(stkRoot, ephemerisPath):
373     for file in os.listdir(ephemerisPath):
374         if file.endswith(".broken"):
375             print("Can't load broken ephemeris. Skipping...")
376         elif file.endswith(".e"):
377             print("Loading file: " + str(file))
378             satellite = stkRoot.CurrentScenario.Children.New(18, file.split(".")[0])
379             satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
380             satellite2.SetPropagatorType(6) # set to STK external (i.e. ephemeris file)
381
382             p = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorStkExternal)
383             p.Filename = os.path.join(ephemerisPath, file)
384
385             try:
386                 p.Propagate()
387             except Exception:
388                 satellite.Unload()
389                 os.rename(os.path.join(ephemerisPath, file), os.path.join(ephemerisPath,
390                                     file.split(".")[0] + ".broken"))
391                 print("Error on propagate, skipping...")
392
393             # Set attitude
394
395             # Set shape/material
396         else:
397             raise Exception("Error. Unexpected file type: " + str(file))
398
399
400 def pointSensorAtSat(sensor, sat):
401     sensor2 = sensor.QueryInterface(STKObjects.IAgSensor)
402     commonTasks = sensor2.CommonTasks
403
404     # 0 = receive, 1 = transmit, 2 = transpond
405     trackMode = 2 # Unclear what this means
406
407     # 0 = Hold, 1 = Level, 2 = Rotate
408     boresightMode = 0 # I don't think this will matter, all of them should point at it
409
410     commonTasks.SetPointingTargetedTracking(trackMode, boresightMode, 'Satellite/' + sat.InstanceName)
411
412
413 def computeAccess(sat, sensor):
414     # Get access by STK Object
415     access = sat.GetAccessToObject(sensor)

```

```

415
416 # Compute access
417 access.ComputeAccess()
418 count = access.ComputedAccessIntervalTimes.Count
419 rtnList = []
420 for i in range(count):
421     rtnList.append(access.ComputedAccessIntervalTimes.GetInterval(i))
422
423 access.Release()
424 return rtnList
425
426
427 def loadEntireNameList():
428     totalNameList = []
429
430     scenarioNameList = ["Script_Scenario_Template_StationKeep_" + str(i) + ".sc" for i in range(1, 6)]
431     for scenarioName in scenarioNameList:
432         satList, nameList, sensor = loadTemplateScenario(stkRoot, scenarioName)
433         totalNameList = totalNameList + nameList
434
435     return totalNameList
436
437
438 if __name__ == "__main__":
439     print("Starting Script...")
440
441     # tleFile = os.path.abspath('test_catalog.tle')
442     tleFile = os.path.abspath('geo-catalog-31july2017.tle')
443     stkGUIVisible = False
444     try:
445         stkRoot, uiApp = startSTK(stkGUIVisible)
446
447         #scenarioName = os.path.join("Script_Scenario_Template_StationKeep", "Script_Scenario_Template_StationKeep.sc")
448         #scenarioName = os.path.join("Script_Scenario_Template_CS0", "Script_Scenario_Template_CS0.sc")
449         #scenarioName = os.path.join("Script_Scenario_Template_Maneuver", "Script_Scenario_Template_Maneuver.sc")
450
451         satList, nameList, sensorList = loadTemplateScenario(stkRoot, scenarioName)
452         scen = stkRoot.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
453
454         # Retrieve Data
455         generateData(scen.StopTime, satList, nameList, sensorList, scenarioName, nameList)
456     finally:
457         stkRoot.CloseScenario
458         del stkRoot
459         del uiApp
460     print("Script Complete. Exiting...")

```

## Appendix C: Keras Experiment

Listing C.1: Keras ANN Implementation and Evaluation Script

```
1 # Title: Keras Experiment
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: This code encapsulates the neural network instantiation, training, and evaluation. It includes k fold
6 # cross validation, though one should still run many replications of it for more significance in obtained results.
7 # Intended to be run with Cluster Experiment in an HPC environment.
8
9 from matplotlib import pyplot as plt
10 import tensorflow as tf
11 from tensorflow.python.keras.callbacks import EarlyStopping, History
12 from tensorflow.python.keras.models import Model, load_model
13 from tensorflow.python.keras.layers import Dense, Dropout, Input, GRU, Lambda, concatenate, Layer, Conv1D, MaxPooling1D,
14     Flatten, GlobalAveragePooling1D
15 from tensorflow.python.keras.optimizers import Adam, RMSprop
16 from tensorflow.python.keras import backend as K
17 from sklearn.model_selection import train_test_split, StratifiedKFold
18 import numpy as np
19 import time
20 from time import sleep
21 import os
22 import pickle
23 import math
24 import sys
25 import csv
26 import argparse
27
28 def build_model(model_name, dropoutRate, opt, numClasses):
29     print("Building model...")
30     lenSoiSignal = 6000
31
32     if model_name == "metric_1":
33         model = build_model_metric(10, dropoutRate, opt, 1)
34     elif model_name == "metric_2":
35         model = build_model_metric(20, dropoutRate, opt, 1)
36     elif model_name == "metric_3":
37         model = build_model_metric(30, dropoutRate, opt, 1)
38     elif model_name == "photo_rnn":
39         model = build_model_photo_rnn(dropoutRate, opt, lenSoiSignal, 1)
40     elif model_name == "photo_convnet":
41         model = build_model_photo_convnet(dropoutRate, opt, lenSoiSignal, 1)
42     elif model_name == "photo_inception":
43         model = build_model_photo_inception(dropoutRate, opt, lenSoiSignal, 1)
44     elif model_name == "both_rnn":
45         model = build_model_both_rnn(dropoutRate, opt, lenSoiSignal, 1)
46     elif model_name == "both_convnet":
47         model = build_model_both_convnet(dropoutRate, opt, lenSoiSignal, 1)
48     elif model_name == "both_inception":
49         model = build_model_both_inception(dropoutRate, opt, lenSoiSignal, 1)
50     else:
51         raise Exception("build_model does not understand model name '" + str(model_name) + "'")
52
53     # Done, so print and return
54     print(model.summary())
55     return model
56
57 def build_model_photo_convnet(dropoutRate=0.0, opt=Adam(), lenSoiSignal=6000, numClasses=2):
58     networkWidth = 64
59
60     # Create the CNN (SOI) portion of the network
61     soiInputs = Input(shape=(lenSoiSignal, 3))
62
63     # Split the light curve from the phase angle
64     lightCurve = Lambda(lambda x: x[:, :, 0:2])(soiInputs)
65     phaseAngle = Lambda(lambda x: x[:, :, 2:3])(soiInputs)
66
67     # Encode the phase angle via the unit circle
68     currLayer = EncodeDegrees()(phaseAngle)
69
70     # Merge phase angle back with the rest of the features
71     currLayer = concatenate([lightCurve, currLayer], axis=-1)
72
73     for i in range(5):
74         currLayer = Conv1D(networkWidth, 3, activation='relu')(currLayer)
75         currLayer = Conv1D(networkWidth, 3, activation='relu')(currLayer)
76         currLayer = MaxPooling1D()(currLayer)
77         currLayer = Flatten()(currLayer)
78
79     # Create output signals
80     for i in range(3):
81         currLayer = Dense(networkWidth, activation='relu')(currLayer)
82         output = Dense(1, activation="sigmoid")(currLayer)
83
84     model = Model(soiInputs, output)
85     model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
86
87     return model
88
89
90 def train_net(model, x_train, y_train, x_val, y_val, maxEpochs, batch_size, patience):
91     print("Beginning training...")
92
93
```

```

94 h = model.fit(x_train, y_train, batch_size=batch_size, epochs=maxEpochs, callbacks=[EarlyStopping(patience=patience)
95 ],
96             validation_data=(x_val, y_val))
97
98 train_loss = h.history['loss'][-1]
99 train_acc = h.history['acc'][-1]
100 val_loss = h.history['val_loss'][-1]
101 val_acc = h.history['val_acc'][-1]
102
103 currEpoch = len(h.history['loss'])
104 return model, train_loss, train_acc, val_loss, val_acc, currEpoch
105
106 def loadData(datadir, seed=1, classNames=[]):
107     # Filter through the available files
108     fileList = os.listdir(datadir)
109     isFirstLoad = True
110     xMetric = []
111     xPhoto = []
112     y = []
113     for filename in fileList:
114         if filename.endswith("_label.npy"):
115             # Check if this is a class we care about
116             if classNames != []:
117                 foundName = False
118                 for className in classNames:
119                     if filename.startswith(className):
120                         foundName = True
121                         break
122             if not foundName:
123                 continue
124
125             baseName = filename.split("_label.npy")[0]
126             print("Loading " + str(baseName))
127             if isFirstLoad:
128                 isFirstLoad = False
129                 xMetric = np.load(os.path.join(datadir, baseName + "_metric.npy"))
130                 xPhoto = np.load(os.path.join(datadir, baseName + "_soi.npy"))
131                 y = np.zeros(xMetric.shape[0])
132             else:
133                 temp = np.load(os.path.join(datadir, baseName + "_metric.npy"))
134                 xMetric = np.concatenate([xMetric, np.load(os.path.join(datadir, baseName + "_metric.npy"))])
135                 xPhoto = np.concatenate([xPhoto, np.load(os.path.join(datadir, baseName + "_soi.npy"))])
136                 y = np.concatenate([y, np.ones(temp.shape[0])])
137
138     # Normalize the data. Note: the time axis must still coincide after normalization, so fix their min/max
139     xMetric[:, :, 0] = normalizeFeature(xMetric[:, :, 0]) # The time axis in the angle measurements
140     xPhoto[:, :, 0] = normalizeFeature(xPhoto[:, :, 0]) # The time axis in the light curve
141     xPhoto[:, :, 1] = normalizeFeature(xPhoto[:, :, 1]) # The effective target irradiance axis in the light curve
142
143     # Remove time information from the light curves
144     xMetric[:, :, 0] = 0
145     xPhoto[:, :, 0] = 0 # The time axis in the light curve
146     xPhoto[:, :, 2] = 0 # The phase angle axis in the light curve
147
148     # Which angle system makes this problem easiest? Stick with pointing.
149     xMetric[:, :, 1] = 0 # Spherical
150     xMetric[:, :, 2] = 0 # Spherical
151     xMetric[:, :, 3] = 0 # Geodetic
152     xMetric[:, :, 4] = 0 # Geodetic
153     xMetric[:, :, 5] = 0 # Pointing
154     xMetric[:, :, 6] = 0 # Pointing
155     xMetric[:, :, 7] = 0 # Phase angle
156
157     return xMetric, xPhoto, y
158
159
160 def normalizeFeature(feature, minVal=[], maxVal=[]):
161     if minVal == []:
162         featureMin = np.min(feature)
163     else:
164         featureMin = minVal
165
166     if maxVal == []:
167         featureMax = np.max(feature)
168     else:
169         featureMax = maxVal
170     return (feature - featureMin) / (featureMax - featureMin)
171
172
173 def log(fileName, message):
174     f = open(fileName, "a")
175     f.writelines([message + "\n"])
176     f.close()
177
178
179 def printTime(seconds):
180     m, s = divmod(seconds, 60)
181     h, m = divmod(m, 60)
182     return "%d Hours %02d Minutes %02d Seconds" % (h, m, s)
183
184
185 def keras_experiment():
186     DEBUG.MODE = True
187
188     # Create the output log
189     outFile = os.path.join(FLAGS.log_dir, FLAGS.log_filename)
190     debugFile = os.path.join(FLAGS.log_dir, FLAGS.log_filename + "_DEBUG")
191     if os.path.exists(outFile):
192         os.remove(outFile)
193
194     if DEBUG.MODE:
195         if os.path.exists(debugFile):
196             os.remove(debugFile)
197
198     try:
199         startTime = time.time()
200         if DEBUG.MODE:

```

```

201     log(debugFile, "Starting the job...")
202     log(debugFile, "rep_num = " + str(FLAGS.rep_num))
203
204     # Input variables for this experiment
205     batch_size = FLAGS.batch_size
206     max_epochs = FLAGS.max_epochs
207     learning_rate = FLAGS.learning_rate
208     dropout_rate = FLAGS.dropout_rate
209     num_layers = FLAGS.num_layers
210     layer_width = FLAGS.layer_width
211     patience = FLAGS.patience
212     rep_num = FLAGS.rep_num
213
214     class_a = FLAGS.class_a
215     class_b = FLAGS.class_b
216
217     training_set = FLAGS.training_set
218     model_type = FLAGS.model_type
219
220     if DEBUG_MODE:
221         log(debugFile, "Listing Inputs Provided...")
222         log(debugFile, "batch_size: " + str(batch_size))
223         log(debugFile, "max_epochs: " + str(max_epochs))
224         log(debugFile, "learning_rate: " + str(learning_rate))
225         log(debugFile, "dropout_rate: " + str(dropout_rate))
226         log(debugFile, "num_layers: " + str(num_layers))
227         log(debugFile, "layer_width: " + str(layer_width))
228         log(debugFile, "patience: " + str(patience))
229         log(debugFile, "training_set: " + str(training_set))
230         log(debugFile, "model_type: " + str(model_type))
231         log(debugFile, "class_a: " + str(class_a))
232         log(debugFile, "class_b: " + str(class_b))
233         log(debugFile, "-----")
234         log(debugFile, "Time Elapsed: " + printTime(time.time() - start_time))
235
236     if class_a != class_b:
237         # Pull the dataset into memory
238         if DEBUG_MODE:
239             log(debugFile, "Loading the dataset into memory")
240             log(debugFile, "Time Elapsed: " + printTime(time.time() - start_time))
241
242         # Load the dataset
243         xMetric, xPhoto, y = loadData(os.path.join(os.environ["WORKDIR"], training_set), classNames=[class_a,
244                                                     class_b])
245
246         if DEBUG_MODE:
247             log(debugFile, "Dataset Info")
248             log(debugFile, "xMetric shape: " + str(xMetric.shape))
249             log(debugFile, "xPhoto shape: " + str(xPhoto.shape))
250             log(debugFile, "y shape: " + str(y.shape))
251             log(debugFile, "-----")
252
253         if model_type.startswith("photo"):
254             # Set the appropriate training/validation/testing data
255             x = xPhoto
256         elif model_type.startswith("metric"):
257             # Set the appropriate training/validation/testing data
258             x = xMetric
259         elif model_type.startswith("both"):
260             # Set the appropriate training/validation/testing data
261             x = [xMetric, xPhoto]
262         else:
263             raise Exception("Error: " + str(model_type) + " is not a valid model choice")
264
265         # Note: change n_splits to 6 for the closely spaced objects experiment
266         skf = StratifiedKFold(n_splits=3, random_state=rep_num, shuffle=True)
267         response_vars = []
268         for train_index, test_index in skf.split(np.zeros(y.shape[0]), y):
269             if model_type.startswith("both"):
270                 x_train_1, x_test_1 = xMetric[train_index, :, :], xMetric[test_index, :, :]
271                 x_train_2, x_test_2 = xPhoto[train_index, :, :], xPhoto[test_index, :, :]
272                 y_train, y_test = y[train_index], y[test_index]
273
274                 # Need some validation data as well
275                 x_train_1, x_val_1, _, _ = train_test_split(x_train_1, y_train, train_size=0.75, random_state=
276                     rep_num)
277                 x_train_2, x_val_2, y_train, y_val = train_test_split(x_train_2, y_train, train_size=0.75,
278                     random_state=rep_num)
279
280                 x_train = [x_train_1, x_train_2]
281                 x_val = [x_val_1, x_val_2]
282                 x_test = [x_test_1, x_test_2]
283             else:
284                 x_train, x_test = x[train_index, :, :], x[test_index, :, :]
285                 y_train, y_test = y[train_index], y[test_index]
286
287                 # Need some validation data as well
288                 x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, train_size=0.75, random_state=
289                     rep_num)
290
291         # Create the model
292         if DEBUG_MODE:
293             log(debugFile, "Creating the model")
294             log(debugFile, "Time Elapsed: " + printTime(time.time() - start_time))
295
296         # Create the model itself
297         model = build_model(model_type, dropout_rate, Adam(lr=learning_rate), 2)
298
299         if DEBUG_MODE:
300             model.summary(print_fn=lambda x: log(debugFile, "Model Summary: " + str(x)))
301
302         # Train the model
303         if DEBUG_MODE:
304             log(debugFile, "Training the model")
305             log(debugFile, "Time Elapsed: " + printTime(time.time() - start_time))
306
307         model, train_loss, train_acc, val_loss, val_acc, numEpochs = train_net(model, x_train, y_train, x_val,
308             y_val, max_epochs, batch_size, patience)

```

```

304         if DEBUG.MODE:
305             log(debugFile, "Training complete, testing...")
306             log(debugFile, "Time Elapsed: " + printTime(time.time() - startTime))
307
308         # Now evaluate on the testing fold
309         metrics = model.evaluate(x_test, y_test, batch_size)
310
311         # Append the resulting accuracies for record
312         response_vars += [str(train_acc), str(metrics[1])]
313
314         total_runtime = time.time() - startTime
315         log(outFile, ",".join(response_vars + [str(numEpochs), str(total_runtime),
316             str(total_runtime / numEpochs)]))
317
318         if DEBUG.MODE:
319             log(debugFile, "Job complete")
320             log(debugFile, "Time Elapsed: " + printTime(time.time() - startTime))
321     else:
322         total_runtime = time.time() - startTime
323         log(outFile, ",".join(3 * ['1.0', '1.0'] + [str(0), str(total_runtime), str(total_runtime / 1)]))
324     return
325 except Exception as excep:
326     log(debugFile, "Exception Occurred: " + str(excep))
327     print("Exception Occurred: " + str(excep))
328
329 def main(_):
330     return keras_experiment()
331
332 if __name__ == '__main__':
333
334     # Instantiate an arg parser.
335     parser = argparse.ArgumentParser()
336
337     # Establish default arguments.
338
339     # These flags are often, but not always, overwritten by the launcher.
340     parser.add_argument('--log_dir', type=str,
341         default='../log/tensorflow_experiment/templog',
342         help='Summaries log directory.')
343
344     parser.add_argument('--log_filename', type=str,
345         default='deep_sa-generalization_experiment.csv',
346         help='Summaries log directory.')
347
348     parser.add_argument('--batch_size', type=int,
349         default=128,
350         help='Training set batch size.')
351
352     parser.add_argument('--max_epochs', type=int,
353         default=1,
354         help='Maximum number of training epochs.')
355
356     parser.add_argument('--learning_rate', type=float, default=1e-4,
357         help='Initial learning rate')
358
359     parser.add_argument('--dropout_rate', type=float,
360         default=0.0,
361         help='Rate at which to dropout neurons.')
362
363     parser.add_argument('--num_layers', type=int,
364         default=1,
365         help='Number of layers in ANN')
366
367     parser.add_argument('--layer_width', type=int,
368         default=100,
369         help='Number of neurons per layer')
370
371     parser.add_argument('--patience', type=int,
372         default=1000000000000000,
373         help='Maximum epochs without improvement before early stopping.')
374
375     parser.add_argument('--training_set', type=str,
376         default="Station_Keeping_Dataset",
377         help='Name of training set to use')
378
379     parser.add_argument('--model_type', type=str,
380         default="both",
381         help='What kind of model to build')
382
383     parser.add_argument('--class_a', type=str,
384         default="Satellite_1",
385         help='Name of first class')
386
387     parser.add_argument('--class_b', type=str,
388         default="Satellite_1",
389         help='Name of second class')
390
391     # These flags specify placekeeping variables.
392     parser.add_argument('--rep_num', type=int,
393         default=0,
394         help='Flag identifying the repetition number.')
395
396     # Parse known arguments.
397     FLAGS, unparsed = parser.parse_known_args()
398
399     # Run the main function as TF app.
400     tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)
401
402

```

## Appendix D: Cluster Experiment

### Listing D.1: Cluster Experiment Framework

```
1 # Title: Cluster Experiment
2 # Author: Capt Justin Fletcher
3 # Organization: Air Force Research Laboratory
4 # Date: 3 August 2018
5 # Description: Framework used to execute code on the HPC. Each experiment will have its own "launcher" that will set
6 # the experimental parameters required. https://github.com/JustinFletcher/hpc-tensorflow
7
8 #!/usr/bin/python
9 # Example PBS cluster job submission in Python
10
11 import os
12 import csv
13 import time
14 import random
15 import argparse
16 import itertools
17 import subprocess
18 import tensorflow as tf
19
20
21 class ClusterExperiment(object):
22
23     def __init__(self):
24
25         self.experimental_configs = []
26         self.independent_designs = []
27         self.coupled_designs = []
28         self.parameter_labels = []
29         self.input_output_maps = []
30         self._job_ids = []
31         self._pbscmd = []
32
33     def add_design(self, flag, value_list):
34
35         self.independent_designs.append((flag, value_list))
36
37     def add_coupled_design(self, coupled_design):
38
39         self.coupled_designs.append(coupled_design)
40
41     def set_rep_count(self, num_reps):
42
43         self.add_design('rep-num', range(num_reps))
44
45     def get_configs(self):
46
47         # Translate the design structure into flag strings.
48         exp_flag_strings = [['--' + f + '=' + str(v) for v in r]
49                             for (f, r) in self.independent_designs]
50
51         # Produce the Cartesian set of configurations.
52         indep_experimental_configs = list(itertools.product(*exp_flag_strings))
53
54         # Initialize a list to store coupled configurations.
55         coupled_configs = []
56
57         # Scope this variable higher due to write-out coupling.
58         coupled_flag_strs = []
59
60         for coupled_design in self.coupled_designs:
61
62             for d in coupled_design:
63
64                 coupled_flag_strs = [['--' + f + '=' + str(v) for v in r]
65                                     for (f, r) in d]
66
67                 coupled_configs += list(itertools.product(*coupled_flag_strs))
68
69         # Initialize empty experimental configs list...
70         experimental_configs = []
71
72         # ...and if there are coupled configs...
73         if coupled_configs:
74
75             # ...iterate over each independent config...
76             for e in indep_experimental_configs:
77
78                 # ...then for each coupled config...
79                 for c in coupled_configs:
80
81                     # ...join the coupled config to the independent one.
82                     experimental_configs.append(e + tuple(c))
83
84         # Otherwise, ....
85         else:
86
87             # ...just pass the independent experiments through.
88             experimental_configs = indep_experimental_configs
89
90         return(experimental_configs)
91
92     def get_parameter_labels(self):
93
94         parameter_labels = []
```

```

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202

for (f, _) in self.independent_designs:
    parameter.labels.append(f)

for coupled_design in self.coupled_designs:
    coupled_flag_strs = []
    for d in coupled_design:
        coupled_flag_strs = [f for (f, _) in d]
    parameter.labels += coupled_flag_strs

return(parameter.labels)

def launch_experiment(self,
    exp_filename,
    log_dir,
    account_str,
    queue_str,
    module_str,
    manager='pbs',
    shuffle_job_order=True):

    experimental_configs = self.get_configs()

    if shuffle_job_order:
        # Shuffle the submission order of configs to avoid asymetries.
        random.shuffle(experimental_configs)

    # Iterate over each experimental configuration, launching jobs.
    for i, experimental_config in enumerate(experimental_configs):
        if manager == 'pbs':
            # Some prints for easy debug.
            print("-----experimental_config-----")
            print(experimental_config)
            print("-----")

            # Customize your options here.
            job_name = "ann_trackid-%d" % i
            walltime = "1:00:00"
            #walltime = "00:15:00"
            select = "1:ncpus=20:mpiprocs=20"
            command = "python " + exp_filename

            # Iterate over flag strings, building the command.
            for flag in experimental_config:
                command += ' ' + flag

            # Add a final flag modifying the log filename to be unique.
            log_filename = 'templog' + str(i)

            command += ' --log_dir=' + log_dir

            # Add the logfile to the command.
            command += ' --log_filename=' + log_filename

            # Build IO maps.
            input_output_map = (experimental_config,
                                log_dir,
                                log_filename)

            self.input_output_maps.append(input_output_map)

            # Build the job string.
            job_string = '#!/bin/bash' + '\n'
            job_string += '#PBS -S /bin/bash' + '\n'
            job_string += '#PBS -N ' + job_name + '\n'
            job_string += '#PBS -l walltime=' + walltime + '\n'
            job_string += '#PBS -l select=' + select + '\n'

            #job_string += '#PBS -joe ' + '\n'
            job_string += '#PBS -o $WORKDIR/log/' + job_name + '.out' + '\n'
            job_string += '#PBS -e $WORKDIR/log/' + job_name + '.err' + '\n'

            job_string += '#PBS -q ' + queue_str + '\n'
            job_string += '#PBS -A ' + account_str + '\n'
            job_string += 'module load ' + module_str + '\n'
            job_string += 'cd $PBS_O_WORKDIR ' + '\n'
            job_string += command

            # Remove below.
            # job_string = ""#!/bin/bash
            # #PBS -N %s
            # #PBS -l walltime=%s
            # #PBS -l select=%s
            # #PBS -o ~/log/output/%s.out
            # #PBS -e ~/log/error/%s.err
            # #PBS -A MHPCC96650DE1
            # #PBS -q standard
            # module load anaconda2/5.0.1 gcc/5.3.0 cudnn/6.0
            # cd $PBS_O_WORKDIR
            # %s"" % (job_name,
            #         walltime,
            #         select,
            #         job_name,
            #         job_name,
            #         command)

            # Print your job string.
            print(job_string)

```

```

203         # Push to file, just in case
204         f = open("pbs_file", "wb")
205         f.write(bytes(job_string))
206         f.close()
207
208         #job_string = bytes(job_string)
209
210         # Use subprocess to command qsub to submit a job.
211         p = subprocess.Popen('qsub',
212                             stdin=subprocess.PIPE,
213                             stdout=subprocess.PIPE,
214                             shell=True)
215
216         # Send job_string to qsub, returning a job ID in bytes.
217         job_id = p.communicate(job_string)[0]
218
219         print(job_id)
220         # Parse the bytes to an ID string.
221         # job_id = str(job_id)
222
223         # Append the job ID.
224         self._job_ids.append(job_id)
225         self._pbscmd.append(job_string)
226
227         print(self._job_ids)
228         time.sleep(0.1)
229
230     else:
231
232         print("Unknown manager supplied to launch_experiment().")
233         exit()
234
235 # qsub -I -A MHPCC96670DA1 -q standard -l select=1:ncpus=20:mpiprocs=20 -l walltime=1:00:00
236
237     def join_job_output(self,
238                        log_dir,
239                        log_filename,
240                        max_runtime,
241                        response_labels):
242
243         jobs_complete = False
244         timeout = False
245         elapsed_time = 0
246
247         # Loop until timeout or all jobs complete.
248         while not(jobs_complete) and not(timeout):
249
250             print("-----")
251             print('Time elapsed: ' + str(elapsed_time) + ' seconds.')
252
253             time.sleep(10)
254
255             elapsed_time += 10
256
257             # Create a list to hold the Bool job complete flags
258             job_complete_flags = []
259
260             # Iterate over each job id string.
261             for job_id in self._job_ids:
262
263                 # TODO: Handle job completion gracefully.
264
265                 # Issue qstat command to get job status.
266                 p = subprocess.Popen('qstat -r ' + job_id,
267                                     stdin=subprocess.PIPE,
268                                     stdout=subprocess.PIPE,
269                                     shell=True)
270
271                 print(job_id)
272
273                 # Get the subprocess output from qstat.
274                 output = p.communicate()
275
276                 # Compute the job completion flag.
277                 try:
278
279                     # Read the qstat out, parse the state, and conv to Boolean.
280                     job_complete = output[0].split()[2] == 'E'
281
282                 except:
283
284                     job_complete = True
285
286                 # Print a diagnostic.
287                 print('Job ' +
288                       job_id +
289                       ' complete? ' +
290                       str(job_complete) +
291                       '\n')
292
293                 # Append the completion flag.
294                 job_complete_flags.append(job_complete)
295
296                 # If the job is complete...
297                 if job_complete:
298
299                     # ...clear it from the queue.
300                     p = subprocess.Popen('qdel -Wforce ' + job_id,
301                                         stdin=subprocess.PIPE,
302                                         stdout=subprocess.PIPE,
303                                         shell=True)
304
305                 # And the job complete flags together.
306                 jobs_complete = (all(job_complete_flags))
307
308                 # Check if we've reached timeout.
309                 timeout = (elapsed_time > max_runtime)
310

```

```

311         if timeout:
312             print("*****")
313             print("TIMEOUT! Terminating jobs...")
314             print("*****")
315
316         # Open a csv for writeout.
317         with open(log_dir + '/' + log_filename, 'w') as csvfile:
318
319             # Join the parameter labels and respons labels, making a header.
320             headers = self.get-parameter-labels() + response_labels
321
322             # Open a writer and write the header.
323             csvwriter = csv.writer(csvfile)
324             csvwriter.writerow(headers)
325
326             # Iterate over each experimental mapping and write out.
327             index = -1
328             for (input_flags,
329                  output_dir,
330                  output_filename) in self._input-output-maps:
331                 index += 1
332                 input_row = []
333
334                 # Process the flags into output values.
335                 for flag in input_flags:
336
337                     flag_val = flag.split('=')[1]
338
339                     input_row.append(flag_val)
340
341                 output_file = os.path.join(output_dir, output_filename)
342
343                 print("Reading output from " + str(output_file))
344
345                 # Check if the output file has been written.
346                 if os.path.exists(output_file):
347
348                     with open(output_file, 'rb') as f:
349
350                         reader = csv.reader(f)
351
352                         for output_row in reader:
353
354                             csvwriter.writerow(input_row + output_row)
355
356                 else:
357                     print("output filename not found: " + output_filename)
358                     if job_complete_flags[index]:
359                         print("this job finished...therefore start it again!")
360
361                     # Use subprocess to command qsub to submit a job.
362                     p = subprocess.Popen('qsub',
363                                           stdin=subprocess.PIPE,
364                                           stdout=subprocess.PIPE,
365                                           shell=True)
366
367                     # Send job_string to qsub, returning a job ID in bytes.
368                     job_id = p.communicate(self._pbscmd[index])[0]
369                     self._job_ids[index] = job_id
370                     jobs_complete = False
371
372             print("-----")
373
374 if __name__ == '__main__':
375
376     # Instantiate an arg parser.
377     parser = argparse.ArgumentParser()
378
379     parser.add_argument('--log_dir', type=str,
380                         default='../log/cluster_experiment/',
381                         help='Summaries log directory.')
382
383     parser.add_argument('--log_filename', type=str,
384                         default='cluster_experiment.csv',
385                         help='Merged output filename.')
386
387     parser.add_argument('--max_runtime', type=int,
388                         default=3600000,
389                         help='Number of seconds to run before giving up.')
390
391     parser.add_argument('--experiment_py_file', type=str,
392                         default='/hpc-tensorflow/cluster_experiment_example.py',
393                         help='Number of seconds to run before giving up.')
394
395     # Parse known arguments.
396     FLAGS, unparsed = parser.parse_known_args()
397
398     example_usage(FLAGS)
399

```

## Appendix E: Process Results

Listing E.1: Python Script to Process Experiment Results

```
1 # Title: Process Results
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: This script processes the results of an HPC run and produces a matrix of the accuracies observed
6 # between RSO combinations. Not used for the closely spaced objects scenario.
7
8 import csv
9 import os
10 import sys
11 import numpy as np
12
13 if __name__ == "__main__":
14     print("-----")
15     print("Starting Script...")
16
17     numFolds = 3
18     numReps = 5
19
20     resultsArr = np.zeros(shape=(8, 8, numFolds * numReps))
21     filename = sys.argv[1]
22     resultsFile = os.path.abspath(filename)
23     outputFile = filename.split(".")[0] + "_processed.csv"
24     with open(resultsFile) as csvfile:
25         reader = csv.reader(csvfile)
26         headerRow = True
27         for row in reader:
28             # Skip the header
29             if headerRow:
30                 headerRow = False
31                 continue
32
33             rep = int(row[0])
34             classAName = row[7]
35             classBName = row[8]
36             classA = int(classAName[-1]) - 1
37             classB = int(classBName[-1]) - 1
38
39             for i in range(numFolds):
40                 resultsArr[classA, classB, (rep * numFolds) + i] = float(row[11 + (2 * i)])
41
42     if np.any(resultsArr == 0):
43         print("Error: Missing at least one rep.")
44         raise Exception("BREAKOUT!")
45
46     meansArr = np.zeros(shape=(8, 8))
47     stdevArr = np.zeros(shape=(8, 8))
48     for classA in range(8):
49         for classB in range(8):
50             # Don't do the lower half of the matrix
51             if classA > classB:
52                 continue
53
54             tempArr = np.concatenate([resultsArr[classA, classB, :], resultsArr[classB, classA, :]])
55             meansArr[classA, classB] = tempArr.mean()
56             stdevArr[classA, classB] = tempArr.std()
57
58     # Now output the results to file
59     with open(outputFile, 'w', newline='') as csvfile:
60         writer = csv.writer(csvfile)
61
62         # First write the means matrix
63         writer.writerow(["Means"])
64         writer.writerow([''] + ["Satellite " + str(i + 1) for i in range(8)])
65         for i in range(8):
66             writer.writerow(["Satellite " + str(i + 1)] + list(meansArr[i, :]))
67
68         writer.writerow([''])
69
70         # Now write the stdev matrix
71         writer.writerow(["Stdev"])
72         writer.writerow([''] + ["Satellite " + str(i + 1) for i in range(8)])
73         for i in range(8):
74             writer.writerow(["Satellite " + str(i + 1)] + list(stdevArr[i, :]))
75
76         # Some more specific metrics...
77         # 1 Difference - Material
78         satList = [(1, 3), (2, 4), (5, 7), (6, 8)]
79         tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
80         writer.writerow(["1 Difference (Material)", tempArr.mean(), "+/-", tempArr.std()])
81
82         # 1 Difference - Shape
83         satList = [(1, 2), (3, 4), (5, 6), (7, 8)]
84         tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
85         writer.writerow(["1 Difference (Shape)", tempArr.mean(), "+/-", tempArr.std()])
86
87         # 1 Difference - Attitude
88         satList = [(1, 5), (2, 6), (3, 7), (4, 8)]
89         tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
90         writer.writerow(["1 Difference (Attitude)", tempArr.mean(), "+/-", tempArr.std()])
91
92         # 2 Differences - Material, Shape
93         satList = [(1, 4), (2, 3), (5, 8), (6, 7)]
94         tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
```

```

95 writer.writerow(["2 Differences (Material, Shape)", tempArr.mean(), "+/-", tempArr.std()])
96
97 # 2 Differences - Material, Attitude
98 satList = [(1, 7), (2, 8), (3, 5), (4, 6)]
99 tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
100 writer.writerow(["2 Differences (Material, Attitude)", tempArr.mean(), "+/-", tempArr.std()])
101
102 # 2 Differences - Shape, Attitude
103 satList = [(1, 6), (2, 5), (3, 8), (4, 7)]
104 tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
105 writer.writerow(["2 Differences (Shape, Attitude)", tempArr.mean(), "+/-", tempArr.std()])
106
107 # 3 Differences
108 satList = [(1, 8), (2, 7), (3, 6), (4, 5)]
109 tempArr = np.concatenate([resultsArr[classA - 1, classB - 1, :] for (classA, classB) in satList])
110 writer.writerow(["3 Differences", tempArr.mean(), "+/-", tempArr.std()])
111
112 print("Script Complete. Exiting...")
113 print("-----")

```

## Appendix F: Baseline Experiment Code

### F.1 Astrogator Script

Listing F.1: Python Script to Automate Execution of Astrogator and Production of RSO Ephemeris Files

```
1 # Title: Run Ephemeris Stationkeep
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Executes STK/Astrogator to generate a station keeping ephemeris for the RSOs in question. When finished,
6 # exports these ephemeris to file.
7
8 from win32com.client import GetActiveObject
9 from comtypes.client import CreateObject
10 from comtypes.gen import STKUtil
11 from comtypes.gen import STKObjects
12 from comtypes.gen import AgStkGatorLib # Must first be generated
13 from sklearn.preprocessing import LabelEncoder
14 from keras.utils import np_utils
15 import win32com
16 import os
17 import numpy as np
18 import time
19 import multiprocessing
20
21 # Only run this line once, possibly even just from a command line. This generates the astrogator module
22 #comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18F4017C}"), 1, 0))
23
24 stkLocation = os.path.join("C:/", "Users", "ianmc", "Documents", "STK 11 (x64)")
25
26
27 def startSTK(isVisible=True):
28     uiApplication = win32com.client.Dispatch('STK11.Application')
29     uiApplication.Visible = isVisible
30
31     # Start new instance of STK
32     uiApplication = CreateObject('STK11.Application')
33     uiApplication.Visible = isVisible
34
35     # Get our IAgStkObjectRoot interface
36     root = uiApplication.Personality2
37
38     # Set some of the units to be a little nicer. I prefer seconds.
39     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
40
41     return root, uiApplication
42
43
44 def createTemplateScenario(root):
45     # IAgStkObjectRoot root: STK Object Model Root
46     root.NewScenario('Script.Scenario.Template')
47
48     # Switch units back temporarily
49     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('UTCG')
50
51     startDatetime = '1 Jan 2019 12:00:00.000'
52     stopDatetime = '1 Jan 2020 12:00:00.000'
53     scen = root.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
54     scen.SetTimePeriod(startDatetime, stopDatetime)
55
56     # And reset the units before we return
57     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
58
59
60 def sanitizeName(name):
61     # Function to remove prohibited characters from a satellite's name
62     badChars = [' ', '(', ')', '/', '\\']
63     rtnName = name
64     for c in badChars:
65         rtnName = rtnName.replace(c, '_')
66
67     return rtnName
68
69
70 def printTime(seconds):
71     m, s = divmod(seconds, 60)
72     h, m = divmod(m, 60)
73     return "%d Hours %02d Minutes %02d Seconds" % (h, m, s)
74
75
76 def getAssignedLongitude(sat):
77     # Grab the average longitude from the first day.
78     # 'Average' should make this work for non-geostationary as well
79     features = ['Detic.LLA.Longitude']
80     timeVals = [i for i in range(0, 60*60*24, 60)]
81     rtnArray = np.zeros(shape=(len(timeVals), len(features)))
82
83     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
84     info = dp.Item('Parameter Set: Cartographic Trajectory').QueryInterface(STKObjects.IAgDataProviderGroup)
85     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
86     prov = group.Item('Trajectory (Cartographic)').QueryInterface(STKObjects.IAgDataPrvTimeVar)
```

```

87     results = prov.ExecSingleElementsArray(timeVals, features)
88
89     for i in range(len(features)):
90         rtnArray[:, i] = results.getArray(i)
91
92     return np.mean(rtnArray[:, 0])
93
94
95 def getMetricData(sat, streakStart, streakEnd, features):
96     # Take a sample at the beginning and end of the streak
97     rtnArray = np.zeros(shape=(1, 2, len(features)))
98
99     # Pull spherical coordinates
100     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
101     info = dp.Item("Parameter Set: Orbit").QueryInterface(STKObjects.IAgDataProviderGroup)
102     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
103     prov = group.Item("Orbit").QueryInterface(STKObjects.IAgDataPrvTimeVar)
104     results = prov.ExecSingleElementsArray([streakStart, streakEnd], features)
105
106     for i in range(len(features)):
107         rtnArray[0, :, i] = results.getArray(i)
108
109     return rtnArray
110
111
112 def addManeuver(targetSequence, dc, maneuverName):
113     dv = targetSequence.Segments.Insert(AgStkGatorLib.eVASegmentTypeManeuver, maneuverName, "-").QueryInterface(
114         AgStkGatorLib.IAgVAMCSManeuver)
115     dv.SetManeuverType(AgStkGatorLib.eVAManeuverTypeImpulsive)
116     impulsive = dv.Maneuver
117     impulsive.SetAttitudeControlType(AgStkGatorLib.eVAAttitudeControlThrustVector)
118
119     # Let the maneuver use all 3 dimensions
120     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianX)
121     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianY)
122     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianZ)
123     xControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.X")
124     yControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Y")
125     zControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Z")
126     xControlParam.Enable = True
127     yControlParam.Enable = True
128     zControlParam.Enable = True
129     xControlParam.MaxStep = 0.1
130     yControlParam.MaxStep = 0.1
131     zControlParam.MaxStep = 0.1
132
133     # Make it a bit easier to read
134     xControlParam.UseCustomDisplayUnit = True
135     yControlParam.UseCustomDisplayUnit = True
136     zControlParam.UseCustomDisplayUnit = True
137     xControlParam.CustomDisplayUnit = "m/sec"
138     yControlParam.CustomDisplayUnit = "m/sec"
139     zControlParam.CustomDisplayUnit = "m/sec"
140
141     dv = dv.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
142     return dv
143
144 def addPropagateByDuration(ts, propName, duration):
145     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
146         IAgVAMCSPropagate)
147     propagate.PropagatorName = "Earth Point Mass"
148     properties = propagate.StoppingConditions["Duration"].Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
149     properties.Trip = duration
150     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
151     return propagate
152
153 def addPropagateToAscendingNode(ts, propName, ascending=True):
154     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
155         IAgVAMCSPropagate)
156     propagate.PropagatorName = "Earth Point Mass"
157     if ascending:
158         properties = propagate.StoppingConditions.Add("AscendingNode").Properties.QueryInterface(AgStkGatorLib.
159             IAgVASToppingCondition)
160     else:
161         properties = propagate.StoppingConditions.Add("DescendingNode").Properties.QueryInterface(AgStkGatorLib.
162             IAgVASToppingCondition)
163     propagate.StoppingConditions.Remove("Duration")
164     properties.CoordSystem = "CentralBody/Earth Fixed"
165     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
166     return propagate
167
168 def addPropagateToApoapsis(ts, propName, apoapsis=True):
169     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
170         IAgVAMCSPropagate)
171     propagate.PropagatorName = "Earth Point Mass"
172     if apoapsis:
173         properties = propagate.StoppingConditions.Add("Apoapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
174     else:
175         properties = propagate.StoppingConditions.Add("Periapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
176     propagate.StoppingConditions.Remove("Duration")
177     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
178     return propagate
179
180 def addNSCorrection(ts, dc):
181     dayLength = 60 * 60 * 24
182
183     # Propagate to ascending node
184     propName = "Propagate to Ascending Node"
185     addPropagateToAscendingNode(ts, propName, ascending=True)
186
187     # Maneuver
188     addManeuver(ts, dc, "Maneuver 1")

```

```

188 # Propagate 0.25 days - check latitude
189 propName = "Propagate for Quarter Day 1"
190 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
191 prop.Results.Add("Geodetic/Latitude")
192 r = dc.Results.GetResultByPaths(propName, "Latitude")
193 r.Enable = True
194 r.DesiredValue = 0.0
195 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
196
197 # Propagate to descending node
198 propName = "Propagate to Descending Node"
199 addPropagateToAscendingNode(ts, propName, ascending=False)
200
201 # Maneuver
202 addManeuver(ts, dc, "Maneuver 2")
203
204 # Propagate 0.25 days - check latitude
205 propName = "Propagate for Quarter Day 2"
206 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
207 prop.Results.Add("Geodetic/Latitude")
208 r = dc.Results.GetResultByPaths(propName, "Latitude")
209 r.Enable = True
210 r.DesiredValue = 0.0
211 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
212
213
214 def addEWTransfer(ts, dc, assignedLong):
215     # Propagate to Apoapsis
216     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
217
218     # Maneuver (no checks)
219     addManeuver(ts, dc, "Maneuver 1")
220
221     # Propagate to Periapasis (shouldn't be needed, just giving another maneuver)
222     addPropagateToApoapsis(ts, "Propagate to Periapsis 1", apoapsis=False)
223
224     # Maneuver (no checks)
225     addManeuver(ts, dc, "Maneuver 2")
226
227     # Propagate to Apoapsis
228     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
229
230     # Maneuver - check periapsis
231     maneuverName = "Maneuver 3"
232     maneuver = addManeuver(ts, dc, maneuverName)
233     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Periapsis")
234     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Periapsis")
235     r.Enable = True
236     r.DesiredValue = 35786.0
237     r.Tolerance = 1.0
238
239     # Propagate to Periapsis
240     addPropagateToApoapsis(ts, "Propagate to Periapsis 2", apoapsis=False)
241
242     # Maneuver - check apoapsis, eccentricity, longitude
243     maneuverName = "Maneuver 4"
244     maneuver = addManeuver(ts, dc, maneuverName)
245     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Apoapsis")
246     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Apoapsis")
247     r.Enable = True
248     r.DesiredValue = 35786.0
249     r.Tolerance = 1.0
250     maneuver.Results.Add("Keplerian Elems/Eccentricity")
251     r = dc.Results.GetResultByPaths(maneuverName, "Eccentricity")
252     r.Enable = True
253     r.DesiredValue = 0.0
254     r.Tolerance = 0.001
255     maneuver.Results.Add("Geodetic/Longitude")
256     r = dc.Results.GetResultByPaths(maneuverName, "Longitude")
257     r.Enable = True
258     r.DesiredValue = assignedLong
259     r.Tolerance = 0.05
260
261
262 def stationKeep(satDriver, stopTime, assignedLong, geostationary=True):
263     if geostationary:
264         # Suppose that each GEO slot is a 0.1 X 0.1 deg square.
265         dayLength = 60 * 60 * 24
266         numDays = 1
267         time = 0
268         dayNum = 0
269         while time < stopTime:
270             # Setup the target sequence and differential corrector
271             ts = satDriver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeTargetSequence, "Day " +
272                 str(dayNum) + ", Station Keeping", "-").QueryInterface(AgStkGatorLib.IAgVAMCSTargetSequence)
273             dc = ts.Profiles.Item(0).QueryInterface(AgStkGatorLib.IAgVAProfileDifferentialCorrector)
274             dc.EnableDisplayStatus = False
275             dc.ClearCorrectionsBeforeRun = True
276             dc.EnableBPlaneNominal = True
277             dc.EnableBPlanePerturbations = True
278             dc.EnableHomotopy = True # Seems to help find solutions progressively.
279             dc.HomotopySteps = 1000
280             dc.RootFindingAlgorithm = 1 # The Newton-Raphson method seems to be more accurate than Secant, but slower
281             dc.Mode = AgStkGatorLib.eVAProfileModeIterate
282             dc.MaxIterations = 200 # Iterations seems to make little difference once Homotopy fails.
283
284             if dayNum == 0:
285                 # Once a year we perform an out-of-plane (e.g. North/South) station-keeping maneuver
286                 addNSCorrection(ts, dc)
287             else:
288                 addEWTransfer(ts, dc, assignedLong)
289
290             # Done with this day
291             time += dayLength * numDays
292             dayNum += 1
293             ts.Action = AgStkGatorLib.eVATargetSeqActionRunActiveProfiles
294         else:
295             # Propagate

```

```

295 propagate = satDriver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypePropagate, "Propagate", "-").
296     QueryInterface(AgStkGatorLib.IAgVAMCSPropagate)
297 propagate.PropagatorName = "Earth Point Mass"
298 properties = propagate.StoppingConditions["Duration"].Properties.QueryInterface(AgStkGatorLib.
299     IAgVAStoppingCondition)
300 properties.Trip = stopTime
301
302 def addTLEWorker(taskList, filepath, ephemerisPath, workerNum):
303     print("Worker " + str(workerNum) + " starting...")
304     numError = 0
305     numWorking = 0
306     numSkipped = 0
307     count = 0
308     jobStartTime = time.time()
309     for task in taskList:
310         name = task[0]
311         ssc = task[1]
312         outFilePath = os.path.join(ephemerisPath, name)
313
314         # Dump the old file (if one exists)
315         if os.path.exists(outFilePath + ".e"):
316             numSkipped += 1
317             count += 1
318             numWorking += 1
319             continue
320         if os.path.exists(outFilePath + ".broken"):
321             os.remove(outFilePath + ".broken")
322
323         # Do the work for this satellite
324         try:
325             # Start new STK instance
326             stkRoot, uiApp = startSTK(False)
327             createTemplateScenario(stkRoot)
328
329             # Ready to load this element into the scenario
330             satellite = stkRoot.CurrentScenario.Children.New(18, name)
331             satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
332
333             satellite2.SetPropagatorType(4) # ePropagatorSGP4
334             propagator = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorSGP4)
335             commonTasks = propagator.CommonTasks.QueryInterface(STKObjects.IAgVePropagatorSGP4CommonTasks)
336             commonTasks.AddSegsFromFile(ssc, filepath)
337             propagator.Propagate()
338
339             # Now pull the ephemeris at scenario start and switch to astrogator
340             x = getMetricData(satellite, 0, 0,
341                 ["Classical.SemimajorAxis", "Classical.Eccentricity", "Classical.Inclination",
342                 "Classical.RAAN", "Classical.ArgumentOfPeriapsis", "Classical.MeanAnomaly"])
343             longitudeSlot = getAssignedLongitude(satellite)
344
345             satellite2.SetPropagatorType(12) # Type 12 is Astrogator
346             driver = satellite2.Propagator.QueryInterface(AgStkGatorLib.IAgVADriverMCS)
347             driver.MainSequence.RemoveAll()
348
349             # Initial State
350             initState = driver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeInitialState, "Initial State",
351                 "-").QueryInterface(AgStkGatorLib.IAgVAMCSInitialState)
352             initState.SetElementType(AgStkGatorLib.eVAElementTypeKeplerian)
353             initState.OrbitEpoch = 0 # Scenario start time
354             kep = initState.Element.QueryInterface(AgStkGatorLib.IAgVAElementKeplerian)
355
356             # For some reason these come back out of order...
357             # Moments like this make me hate STK quite deeply
358             kep.SemiMajorAxis = x[0, 0, 4]
359             kep.Eccentricity = x[0, 0, 0]
360             kep.Inclination = x[0, 0, 1]
361             kep.RAAN = x[0, 0, 2]
362             kep.ArgOfPeriapsis = x[0, 0, 3]
363             kep.MeanAnomaly = x[0, 0, 5]
364
365             scen = stkRoot.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
366
367             stationKeep(driver, scen.StopTime, longitudeSlot, geostationary=True)
368             driver.Options.ClearDWCGraphicsBeforeEachRun = True
369             errorCode = driver.RunMCS2()
370
371             # Check if it worked
372             if errorCode != 0:
373                 satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".broken")
374                 numError += 1
375             else:
376                 satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".e")
377                 numWorking += 1
378             satellite.Unload()
379
380             count += 1
381             timeElapsed = time.time() - jobStartTime
382             rate = (count - numSkipped) / timeElapsed
383             print("-----")
384             print("Worker " + str(workerNum) + ": " + str(count) + " of " + str(len(taskList)) + " done.")
385             print("Worker " + str(workerNum) + ": " + str(numWorking) + " of " + str(len(taskList)) + " working.")
386             print("Worker " + str(workerNum) + ": " + str(numError) + " of " + str(len(taskList)) + " errors.")
387             print("Worker " + str(workerNum) + ": " + str(100 * count / len(taskList)) + "% complete.")
388             print("Worker " + str(workerNum) + ": " + printTime(timeElapsed) + " elapsed")
389             print("Worker " + str(workerNum) + ": " + str(rate * 60) + " tasks per minute")
390             print("Worker " + str(workerNum) + ": " + printTime((
391                 len(taskList) - count) / rate) + " est time remaining")
392             print("-----")
393         finally:
394             # Done, so return and kill this STK instance
395             stkRoot.CloseScenario
396             del stkRoot
397             del uiApp
398     print("Worker " + str(workerNum) + " complete. Exiting...")
399

```

```

400 def addTLEFile(filepath, ephemerisPath, numWorkers=1):
401
402     # Assemble the list of tasks to assign to each worker
403     taskList = []
404
405     f = open(filepath, 'r')
406     filelines = f.readlines()
407     f.close()
408
409     # In my case, every line of the TLE starts with the line number
410     # I have seen examples where line 0 doesn't, but hope that doesn't
411     # happen in our set.
412     for line in filelines:
413         if line.startswith('0'):
414             name = sanitizeName(line[2:-1])
415         elif line.startswith('1'):
416             # Because someone thought it would be a good idea to stick classification on the end with no whitespace...
417             number = line.split()[1]
418             ssc = number[:-1]
419         elif line.startswith('2') and (float(line.split()[2]) < 1.0):
420             taskList.append([name, ssc])
421
422     # Pass out the tasks to each worker
423     count = 0
424     workerList = [[] for i in range(numWorkers)]
425     for task in taskList:
426         workerList[count % numWorkers].append(task)
427         count += 1
428
429     # Push these jobs off to multiprocessing
430     jobs = []
431     for i in range(numWorkers):
432         if workerList[i]:
433             p = multiprocessing.Process(target=addTLEWorker, args=(workerList[i], filepath, ephemerisPath, i))
434             jobs.append(p)
435             p.start()
436
437     # Wait for them all to finish
438     for j in jobs:
439         j.join()
440
441
442 if __name__ == "__main__":
443     print("Starting Script...")
444     tleFile = os.path.abspath('geo_catalog_31july2017.tle')
445     numWorker = 4
446
447     addTLEFile(os.path.abspath(tleFile), os.path.abspath('Ephemeris_StationKeeping'), numWorker)
448
449     print("Script Complete. Exiting...")

```

## F.2 Experiment Launcher

Listing F.2: Cluster Experiment Launcher for Baseline Experiment

```

1  # Title: Launcher for Station Keeping Experiment
2  # Author: 1stLt Ian McQuaid
3  # Organization: Air Force Institute of Technology
4  # Date: 3 August 2018
5  # Description: Launches the station keeping experiment on the HPC. This uses the cluster experiment framework, along
6  # with the keras experiment ANN code to run many machine learning models over a short period of time. This experiment
7  # examines all pairs of RSOs.
8
9  #!/usr/bin/python
10 # Example PBS cluster job submission in Python
11
12 import argparse
13 import os
14 import tensorflow as tf
15
16 from cluster_experiment import ClusterExperiment
17
18 def main(FLAGS):
19
20     # Clear and remake the log directory.
21     if tf.gfile.Exists(FLAGS.log_dir):
22         tf.gfile.DeleteRecursively(FLAGS.log_dir)
23
24     tf.gfile.MakeDirs(FLAGS.log_dir)
25
26     # Instantiate an experiment.
27     exp = ClusterExperiment()
28
29     # Set the number of reps for each config.
30     exp.set_rep_count(5)
31
32     # Set independent parameters.
33     exp.add.design('batch-size', [128])
34     exp.add.design('max-epochs', [10000000000])
35     exp.add.design('learning-rate', [0.00001])
36     exp.add.design('dropout-rate', [0.0])
37     exp.add.design('patience', [30])
38     exp.add.design('training-set', ["Station-Keeping-Dataset"])
39     exp.add.design('class-a', ['Satellite-1', 'Satellite-2', 'Satellite-3', 'Satellite-4', 'Satellite-5', 'Satellite-6',
40     'Satellite-7', 'Satellite-8'])
41     exp.add.design('class-b', ['Satellite-1', 'Satellite-2', 'Satellite-3', 'Satellite-4', 'Satellite-5', 'Satellite-6',
42     'Satellite-7', 'Satellite-8'])
43
44

```

```

45 exp.add.design('model.type', ["photo-convnet"])
46
47 # Launch the experiment.
48 exp.launch_experiment(exp.filename=FLAGS.experiment_py_file,
49                       log_dir=FLAGS.log_dir,
50                       account_str='MHPCC96670DA1',
51                       queue_str='standard',
52                       module_str='anaconda2/5.0.1 gcc/5.3.0 cudnn/6.0 tensorflow/1.8.0',
53                       manager='pbs',
54                       shuffle_job_order=True)
55
56 # Manually note response variables.
57 response_labels = ['train_acc_1',
58                   'test_acc_1',
59                   'train_acc_2',
60                   'test_acc_2',
61                   'train_acc_3',
62                   'test_acc_3',
63                   'num_epochs',
64                   'total_runtime',
65                   'time_per_epoch',]
66
67 # Wait for the output to return.
68 exp.join_job_output(FLAGS.log_dir,
69                    FLAGS.log_filename,
70                    FLAGS.max_runtime,
71                    response_labels)
72
73 print("All jobs complete. Exiting.")
74
75 if __name__ == '__main__':
76
77     # Instantiate an arg parser.
78     parser = argparse.ArgumentParser()
79
80     parser.add_argument('--log_dir', type=str,
81                       default=os.path.join(os.environ["WORKDIR"], 'log'),
82                       help='Summaries log directory.')
83
84     parser.add_argument('--log_filename', type=str,
85                       default='experiment.results.csv',
86                       help='Merged output filename.')
87
88     parser.add_argument('--max_runtime', type=int,
89                       default=3600000,
90                       help='Number of seconds to run before giving up.')
91
92     parser.add_argument('--experiment_py_file', type=str,
93                       default=os.path.join(os.environ["WORKDIR"], 'experiment-photo-pairwise-stationkeep', '
94                                           keras.experiment.py'),
95                       help='Name of the experiment file.')
96
97     # Parse known arguments.
98     FLAGS, unparsed = parser.parse_known_args()
99
100    main(FLAGS)

```

## Appendix G: Closely Spaced Objects Experiment Code

### G.1 Astrogator Script

Listing G.1: Python Script to Automate Execution of Astrogator and Production of RSO Ephemeris Files

```
1 # Title: Run Ephemeris Closely Spaced Objects
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Executes STK/Astrogator to generate a station keeping ephemeris for the RSOs in question. Here the
6 # stations are assigned to match the desired closely spaced object's scenario. When finished, exports these ephemeris
7 # to file.
8
9 from win32com.client import GetActiveObject
10 from comtypes.client import CreateObject
11 from comtypes.gen import STKUtil
12 from comtypes.gen import STKObjects
13 from comtypes.gen import AgStkGatorLib # Must first be generated
14 from sklearn.preprocessing import LabelEncoder
15 from keras.utils import np_utils
16 import win32com
17 import os
18 import numpy as np
19 import time
20 import multiprocessing
21
22 # Only run this line once, possibly even just from a command line. This generates the astrogator module
23 #comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18F4017C}"), 1, 0))
24
25 stkLocation = os.path.join("C:/", "Users", "ianmc", "Documents", "STK 11 (x64)")
26
27
28 def startSTK(isVisible=True):
29     uiApplication = win32com.client.Dispatch('STK11.Application')
30     uiApplication.Visible = isVisible
31
32     # Start new instance of STK
33     uiApplication = CreateObject('STK11.Application')
34     uiApplication.Visible = isVisible
35
36     # Get our IAgStkObjectRoot interface
37     root = uiApplication.Personality2
38
39     # Set some of the units to be a little nicer. I prefer seconds.
40     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
41
42     return root, uiApplication
43
44
45 def createTemplateScenario(root):
46     # IAgStkObjectRoot root: STK Object Model Root
47     root.NewScenario('Script.Scenario.Template')
48
49     # Switch units back temporarily
50     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('UTC')
51
52     startDatetime = '1 Jan 2019 12:00:00.000'
53     stopDatetime = '1 Jan 2020 12:00:00.000'
54     scen = root.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
55     scen.SetTimePeriod(startDatetime, stopDatetime)
56
57     # And reset the units before we return
58     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
59
60
61 def sanitizeName(name):
62     # Function to remove prohibited characters from a satellite's name
63     badChars = [' ', '(', ')', '/', '\\']
64     rtnName = name
65     for c in badChars:
66         rtnName = rtnName.replace(c, '_')
67
68     return rtnName
69
70
71 def printTime(seconds):
72     m, s = divmod(seconds, 60)
73     h, m = divmod(m, 60)
74     return "%d Hours %02d Minutes %02d Seconds" % (h, m, s)
75
76
77 def getAssignedLongitude(sat):
78     # Grab the average longitude from the first day.
79     # 'Average' should make this work for non-geostationary as well
80     features = ['Detic.LLA.Longitude']
81     timeVals = [i for i in range(0, 60*60*24, 60)]
82     rtnArray = np.zeros(shape=(len(timeVals), len(features)))
83
84     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
85     info = dp.Item('Parameter Set: Cartographic Trajectory').QueryInterface(STKObjects.IAgDataProviderGroup)
86     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
```

```

87 prov = group.Item('Trajectory (Cartographic)').QueryInterface(STKObjects.IAgDataPrvTimeVar)
88 results = prov.ExecSingleElementsArray(timeVals, features)
89
90 for i in range(len(features)):
91     rtnArray[:, i] = results.GetArray(i)
92
93 return np.mean(rtnArray[:, 0])
94
95
96 def getMetricData(sat, streakStart, streakEnd, features):
97     # Take a sample at the beginning and end of the streak
98     rtnArray = np.zeros(shape=(1, 2, len(features)))
99
100     # Pull spherical coordinates
101     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
102     info = dp.Item('Parameter Set: Orbit').QueryInterface(STKObjects.IAgDataProviderGroup)
103     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
104     prov = group.Item('Orbit').QueryInterface(STKObjects.IAgDataPrvTimeVar)
105     results = prov.ExecSingleElementsArray([streakStart, streakEnd], features)
106
107     for i in range(len(features)):
108         rtnArray[0, :, i] = results.GetArray(i)
109
110     return rtnArray
111
112
113 def addManeuver(targetSequence, dc, maneuverName):
114     dv = targetSequence.Segments.Insert(AgStkGatorLib.eVASegmentTypeManeuver, maneuverName, "-").QueryInterface(
115         AgStkGatorLib.IAgVAMCSManeuver)
116     dv.SetManeuverType(AgStkGatorLib.eVAManeuverTypeImpulsive)
117     impulsive = dv.Maneuver
118     impulsive.SetAttitudeControlType(AgStkGatorLib.eVAAttitudeControlThrustVector)
119
120     # Let the maneuver use all 3 dimensions
121     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianX)
122     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianY)
123     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianZ)
124     xControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.X")
125     yControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Y")
126     zControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Z")
127     xControlParam.Enable = True
128     yControlParam.Enable = True
129     zControlParam.Enable = True
130     xControlParam.MaxStep = 0.1
131     yControlParam.MaxStep = 0.1
132     zControlParam.MaxStep = 0.1
133
134     # Make it a bit easier to read
135     xControlParam.UseCustomDisplayUnit = True
136     yControlParam.UseCustomDisplayUnit = True
137     zControlParam.UseCustomDisplayUnit = True
138     xControlParam.CustomDisplayUnit = "m/sec"
139     yControlParam.CustomDisplayUnit = "m/sec"
140     zControlParam.CustomDisplayUnit = "m/sec"
141
142     dv = dv.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
143     return dv
144
145 def addPropagateByDuration(ts, propName, duration):
146     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
147         IAgVAMCSPropagate)
148     propagate.PropagatorName = "Earth Point Mass"
149     properties = propagate.StoppingConditions["Duration"].Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
150     properties.Trip = duration
151     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
152     return propagate
153
154 def addPropagateToAscendingNode(ts, propName, ascending=True):
155     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
156         IAgVAMCSPropagate)
157     propagate.PropagatorName = "Earth Point Mass"
158     if ascending:
159         properties = propagate.StoppingConditions.Add("AscendingNode").Properties.QueryInterface(AgStkGatorLib.
160             IAgVASToppingCondition)
161     else:
162         properties = propagate.StoppingConditions.Add("DescendingNode").Properties.QueryInterface(AgStkGatorLib.
163             IAgVASToppingCondition)
164     propagate.StoppingConditions.Remove("Duration")
165     properties.CoordSystem = "CentralBody/Earth Fixed"
166     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
167     return propagate
168
169 def addPropagateToApoapsis(ts, propName, apoapsis=True):
170     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
171         IAgVAMCSPropagate)
172     propagate.PropagatorName = "Earth Point Mass"
173     if apoapsis:
174         properties = propagate.StoppingConditions.Add("Apoapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
175     else:
176         properties = propagate.StoppingConditions.Add("Periapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
177     propagate.StoppingConditions.Remove("Duration")
178     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
179     return propagate
180
181 def addNSCorrection(ts, dc):
182     dayLength = 60 * 60 * 24
183
184     # Propagate to ascending node
185     propName = "Propagate to Ascending Node"
186     addPropagateToAscendingNode(ts, propName, ascending=True)
187
188     # Maneuver
189     addManeuver(ts, dc, "Maneuver 1")

```

```

188
189 # Propagate 0.25 days - check latitude
190 propName = "Propagate for Quarter Day 1"
191 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
192 prop.Results.Add("Geodetic/Latitude")
193 r = dc.Results.GetResultByPaths(propName, "Latitude")
194 r.Enable = True
195 r.DesiredValue = 0.0
196 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
197
198 # Propagate to descending node
199 propName = "Propagate to Descending Node"
200 addPropagateToAscendingNode(ts, propName, ascending=False)
201
202 # Maneuver
203 addManeuver(ts, dc, "Maneuver 2")
204
205 # Propagate 0.25 days - check latitude
206 propName = "Propagate for Quarter Day 2"
207 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
208 prop.Results.Add("Geodetic/Latitude")
209 r = dc.Results.GetResultByPaths(propName, "Latitude")
210 r.Enable = True
211 r.DesiredValue = 0.0
212 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
213
214
215 def addEWTransfer(ts, dc, assignedLong):
216     # Propagate to Apoapsis
217     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
218
219     # Maneuver (no checks)
220     addManeuver(ts, dc, "Maneuver 1")
221
222     # Propagate to Periapsis (shouldn't be needed, just giving another maneuver)
223     addPropagateToApoapsis(ts, "Propagate to Periapsis 1", apoapsis=False)
224
225     # Maneuver (no checks)
226     addManeuver(ts, dc, "Maneuver 2")
227
228     # Propagate to Apoapsis
229     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
230
231     # Maneuver - check periapsis
232     maneuverName = "Maneuver 3"
233     maneuver = addManeuver(ts, dc, maneuverName)
234     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Periapsis")
235     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Periapsis")
236     r.Enable = True
237     r.DesiredValue = 35786.0
238     r.Tolerance = 1.0
239
240     # Propagate to Periapsis
241     addPropagateToApoapsis(ts, "Propagate to Periapsis 2", apoapsis=False)
242
243     # Maneuver - check apoapsis, eccentricity, longitude
244     maneuverName = "Maneuver 4"
245     maneuver = addManeuver(ts, dc, maneuverName)
246     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Apoapsis")
247     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Apoapsis")
248     r.Enable = True
249     r.DesiredValue = 35786.0
250     r.Tolerance = 1.0
251     maneuver.Results.Add("Keplerian Elems/Eccentricity")
252     r = dc.Results.GetResultByPaths(maneuverName, "Eccentricity")
253     r.Enable = True
254     r.DesiredValue = 0.0
255     r.Tolerance = 0.001
256     maneuver.Results.Add("Geodetic/Longitude")
257     r = dc.Results.GetResultByPaths(maneuverName, "Longitude")
258     r.Enable = True
259     r.DesiredValue = assignedLong
260     r.Tolerance = 0.05
261
262
263 def stationKeep(satDriver, stopTime, assignedLong, geostationary=True):
264     if geostationary:
265         # Suppose that each GEO slot is a 0.1 X 0.1 deg square.
266         dayLength = 60 * 60 * 24
267         numDays = 1
268         time = 0
269         dayNum = 0
270         while time < stopTime:
271             # Setup the target sequence and differential corrector
272             ts = satDriver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeTargetSequence, "Day " +
273                 str(dayNum) + ": Station Keeping", "-").QueryInterface(AgStkGatorLib.IAgVAMCSTargetSequence)
274             dc = ts.Profiles.Item(0).QueryInterface(AgStkGatorLib.IAgVAProfileDifferentialCorrector)
275             dc.EnableDisplayStatus = False
276             dc.ClearCorrectionsBeforeRun = True
277             dc.EnableBPlaneNominal = True
278             dc.EnableBPlanePerturbations = True
279             dc.EnableHomotopy = True # Seems to help find solutions progressively.
280             dc.HomotopySteps = 1000
281             dc.RootFindingAlgorithm = 1 # The Newton-Raphason method seems to be more accurate than Secant, but slower
282             dc.Mode = AgStkGatorLib.eVAProfileModelIterate
283             dc.MaxIterations = 200 # Iterations seems to make little difference once Homotopy fails.
284
285             if dayNum == 0:
286                 # Once a year we perform an out-of-plane (e.g. North/South) station-keeping maneuver
287                 addNSCorrection(ts, dc)
288             else:
289                 addEWTransfer(ts, dc, assignedLong)
290
291             # Done with this day
292             time += dayLength * numDays
293             dayNum += 1
294             ts.Action = AgStkGatorLib.eVATargetSeqActionRunActiveProfiles
295         else:

```

```

295     # Propagate
296     propagate = satDriver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypePropagate, "Propagate", "-").
        QueryInterface(AgStkGatorLib.IAgVAMCSPropagate)
297     propagate.PropagatorName = "Earth Point Mass"
298     properties = propagate.StoppingConditions["Duration"].Properties.QueryInterface(AgStkGatorLib.
        IAgVAStoppingCondition)
299     properties.Trip = stopTime
300
301
302 def addTLEWorker(taskList, filepath, ephemerisPath, workerNum):
303     print("Worker " + str(workerNum) + " starting...")
304
305     numError = 0
306     numWorking = 0
307     numSkipped = 0
308     count = 0
309     jobStartTime = time.time()
310     for task in taskList:
311         name = task[0]
312         ssc = task[1]
313         longitudeSlot = task[2]
314         outFilePath = os.path.join(ephemerisPath, name)
315
316         # Dump the old file (if one exists)
317         if os.path.exists(outFilePath + ".e"):
318             numSkipped += 1
319             count += 1
320             numWorking += 1
321             continue
322         if os.path.exists(outFilePath + ".broken"):
323             os.remove(outFilePath + ".broken")
324
325         # Do the work for this satellite
326         try:
327             # Start new STK instance
328             stkRoot, uiApp = startSTK(False)
329             createTemplateScenario(stkRoot)
330
331             # Ready to load this element into the scenario
332             satellite = stkRoot.CurrentScenario.Children.New(18, name)
333             satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
334
335             satellite2.SetPropagatorType(4) # ePropagatorSGP4
336             propagator = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorSGP4)
337             commonTasks = propagator.CommonTasks.QueryInterface(STKObjects.IAgVePropagatorSGP4CommonTasks)
338             commonTasks.AddSegsFromFile(ssc, filepath)
339             propagator.Propagate()
340
341             # Now pull the ephemeris at scenario start and switch to astrogator
342             x = getMetricData(satellite, 0, 0,
343                             ["Classical.SemimajorAxis", "Classical.Eccentricity", "Classical.Inclination",
344                             "Classical.RAAN", "Classical.ArgumentOfPeriapsis", "Classical.MeanAnomaly"])
345             #longitudeSlot = getAssignedLongitude(satellite)
346
347             satellite2.SetPropagatorType(12) # Type 12 is Astrogator
348             driver = satellite2.Propagator.QueryInterface(AgStkGatorLib.IAgVADriverMCS)
349             driver.MainSequence.RemoveAll()
350
351             # Initial State
352             initState = driver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeInitialState, "Initial State",
353                                                     "-").QueryInterface(AgStkGatorLib.IAgVAMCSInitialState)
354             initState.SetElementType(AgStkGatorLib.eVAElementTypeKeplerian)
355             initState.OrbitEpoch = 0 # Scenario start time
356             kep = initState.Element.QueryInterface(AgStkGatorLib.IAgVAElementKeplerian)
357
358             # For some reason these come back out of order...
359             # Moments like this make me hate STK quite deeply
360             kep.SemiMajorAxis = x[0, 0, 4]
361             kep.Eccentricity = x[0, 0, 0]
362             kep.Inclination = x[0, 0, 1]
363             kep.RAAN = x[0, 0, 2]
364             kep.ArgOfPeriapsis = x[0, 0, 3]
365             kep.MeanAnomaly = x[0, 0, 5]
366
367             scen = stkRoot.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
368
369             stationKeep(driver, scen.StopTime, longitudeSlot, geostationary=True)
370             driver.Options.ClearDWCGraphicsBeforeEachRun = True
371             errorCode = driver.RunMCS2()
372
373             # Check if it worked
374             if errorCode != 0:
375                 satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".broken")
376                 numError += 1
377             else:
378                 satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".e")
379                 numWorking += 1
380             satellite.Unload()
381
382             count += 1
383             timeElapsed = time.time() - jobStartTime
384             rate = (count - numSkipped) / timeElapsed
385             print("Worker " + str(workerNum) + ": " + str(count) + " of " + str(len(taskList)) + " done.")
386             print("Worker " + str(workerNum) + ": " + str(numWorking) + " of " + str(len(taskList)) + " working.")
387             print("Worker " + str(workerNum) + ": " + str(numError) + " of " + str(len(taskList)) + " errors.")
388             print("Worker " + str(workerNum) + ": " + str(100 * count / len(taskList)) + "% complete.")
389             print("Worker " + str(workerNum) + ": " + printTime(timeElapsed) + " elapsed")
390             print("Worker " + str(workerNum) + ": " + str(rate * 60) + " tasks per minute")
391             print("Worker " + str(workerNum) + ": " + printTime((
392                 len(taskList) - count) / rate) + " est time remaining")
393         print("-----")
394     finally:
395         # Done, so return and kill this STK instance
396         stkRoot.CloseScenario
397         del stkRoot
398         del uiApp
399     print("Worker " + str(workerNum) + " complete. Exiting...")

```

```

400
401
402 def addTLEFile(filepath, ephemerisPath, satNames, satLongs, numWorkers=1):
403
404     # Assemble the list of tasks to assign to each worker
405     taskList = []
406
407     f = open(filepath, 'r')
408     filelines = f.readlines()
409     f.close()
410
411     # In my case, every line of the TLE starts with the line number
412     # I have seen examples where line 0 doesn't, but hope that doesn't
413     # happen in our set.
414     for line in filelines:
415         if line.startswith('0'):
416             name = sanitizeName(line[2:-1])
417             elif line.startswith('1'):
418                 # Because someone thought it would be a good idea to stick classification on the end with no whitespace...
419                 number = line.split()[1]
420                 ssc = number[:-1]
421             elif line.startswith('2') and (float(line.split()[2]) < 1.0):
422                 if name in satNames:
423                     index = satNames.index(name)
424                     taskList.append([name, ssc, satLongs[index]])
425
426     # Pass out the tasks to each worker
427     count = 0
428     workerList = [[] for i in range(numWorkers)]
429     for task in taskList:
430         workerList[count % numWorkers].append(task)
431         count += 1
432
433     # Push these jobs off to multiprocessing
434     jobs = []
435     for i in range(numWorkers):
436         if workerList[i]:
437             p = multiprocessing.Process(target=addTLEWorker, args=(workerList[i], filepath, ephemerisPath, i))
438             jobs.append(p)
439             p.start()
440
441     # Wait for them all to finish
442     for j in jobs:
443         j.join()
444
445 if __name__ == "__main__":
446     print("Starting Script...")
447     tleFile = os.path.abspath('geo-catalog-31july2017.tle')
448     numWorker = 4
449
450     namesToSearch = ["JCSAT-16", "INTELSAT-907", "GOES-13", "AMC-16", "GOES-16", "DIRECTV-11", "MEXSAT-3", "GALAXY-15"]
451     stations = [-150.0, -150.01, -150.1, -150.2, -150.3, -150.4, -150.5, -150.6]
452
453     addTLEFile(os.path.abspath(tleFile), os.path.abspath('Ephemeris-Custom'), namesToSearch, stations, numWorker)
454
455     print("Script Complete. Exiting...")
456

```

## G.2 Experiment Launcher

### Listing G.2: Cluster Experiment Launcher for Closely Spaced Objects Experiment

```

1 # Title: Launcher for Closely Spaced Objects Experiment
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Launches the closely spaced objects experiment on the HPC. This uses the cluster experiment framework,
6 # along with the keras experiment ANN code to run many machine learning models over a short period of time. This
7 # experiment only compares satellite 1 against the rest of the catalog.
8
9 #!/usr/bin/python
10 # Example PBS cluster job submission in Python
11
12 import argparse
13 import os
14 import tensorflow as tf
15
16 # import pyhpc as hpc
17 # hpc.ClusterExperiment()
18
19 from cluster_experiment import ClusterExperiment
20
21
22 def main(FLAGS):
23
24     # Clear and remake the log directory.
25     if tf.gfile.Exists(FLAGS.log_dir):
26
27         tf.gfile.DeleteRecursively(FLAGS.log_dir)
28
29     tf.gfile.MakeDirs(FLAGS.log_dir)
30
31     # Instantiate an experiment.
32     exp = ClusterExperiment()
33
34     # Set the number of reps for each config.
35     exp.set_rep_count(5)
36
37     # Set independent parameters.

```

```

38 exp.add_design('batch-size', [32])
39 exp.add_design('max_epochs', [10000000000])
40 exp.add_design('learning-rate', [0.00001])
41 exp.add_design('dropout-rate', [0.0])
42 exp.add_design('patience', [30])
43 exp.add_design('training-set', ["CSO-Dataset-New"])
44 exp.add_design('class-a', ['Satellite-1'])
45 exp.add_design('class-b', ['Satellite-2', 'Satellite-3', 'Satellite-4', 'Satellite-5', 'Satellite-6',
46                               'Satellite-7', 'Satellite-8'])
47
48 exp.add_design('model-type', ["photo-convnet"])
49
50 # Launch the experiment.
51 exp.launch_experiment(exp_filename=FLAGS.experiment_py_file,
52                       log_dir=FLAGS.log_dir,
53                       account_str='MHPCC96670DA1',
54                       queue_str='standard',
55                       module_str='anaconda2/5.0.1 gcc/5.3.0 cudnn/6.0 tensorflow/1.8.0',
56                       manager='pbs',
57                       shuffle_job_order=True)
58
59 # Manually note response variables.
60 response_labels = ['train_acc_1',
61                   'test_acc_1',
62                   'train_acc_2',
63                   'test_acc_2',
64                   'train_acc_3',
65                   'test_acc_3',
66                   'train_acc_4',
67                   'test_acc_4',
68                   'train_acc_5',
69                   'test_acc_5',
70                   'train_acc_6',
71                   'test_acc_6',
72                   'num_epochs',
73                   'total_runtime',
74                   'time_per_epoch',]
75
76 # Wait for the output to return.
77 exp.join_job_output(FLAGS.log_dir,
78                    FLAGS.log_filename,
79                    FLAGS.max_runtime,
80                    response_labels)
81
82 print("All jobs complete. Exiting.")
83
84 if __name__ == '__main__':
85
86     # Instantiate an arg parser.
87     parser = argparse.ArgumentParser()
88
89     parser.add_argument('--log_dir', type=str,
90                         default=os.path.join(os.environ["WORKDIR"], 'log'),
91                         help='Summaries log directory.')
92
93     parser.add_argument('--log_filename', type=str,
94                         default='experiment_results.csv',
95                         help='Merged output filename.')
96
97     parser.add_argument('--max_runtime', type=int,
98                         default=3600000,
99                         help='Number of seconds to run before giving up.')
100
101     parser.add_argument('--experiment_py_file', type=str,
102                         default=os.path.join(os.environ["WORKDIR"], 'experiment-photo-cso', 'keras_experiment.py'),
103                         help='Name of the experiment file.')
104
105     # Parse known arguments.
106     FLAGS, unparsed = parser.parse_known_args()
107
108     main(FLAGS)
109

```

## Appendix H: Maneuvering Experiment Code

### H.1 Astrogator Script

Listing H.1: Python Script to Automate Execution of Astrogator and Production of RSO Ephemeris Files

```
1 # Title: Run Ephemeris Maneuvering
2 # Author: 1stLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Executes STK/Astrogator to generate a maneuvering ephemeris for the RSOs in question. Usual stations are
6 # initially assigned, however RSOs permute stations approximately ever 31 days. The order they permute is based on the
7 # RSOs order in the original TLE file. When finished, exports these ephemeris to file.
8
9 from win32com.client import GetActiveObject
10 from comtypes.client import CreateObject
11 from comtypes.gen import STKUtil
12 from comtypes.gen import STKObjects
13 from comtypes.gen import AgStkGatorLib # Must first be generated
14 from sklearn.preprocessing import LabelEncoder
15 from keras.utils import np_utils
16 import win32com
17 import os
18 import numpy as np
19 import time
20 import sys
21 import multiprocessing
22
23 # Only run this line once, possibly even just from a command line. This generates the astrogator module
24 #comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18F4017C}"), 1, 0))
25
26 stkLocation = os.path.join("C:/", "Users", "ianme", "Documents", "STK 11 (x64)")
27
28
29 def startSTK(isVisible=True):
30     uiApplication = win32com.client.Dispatch('STK11.Application')
31     uiApplication.Visible = isVisible
32
33     # Start new instance of STK
34     uiApplication = CreateObject('STK11.Application')
35     uiApplication.Visible = isVisible
36
37     # Get our IAgStkObjectRoot interface
38     root = uiApplication.Personality2
39
40     # Set some of the units to be a little nicer. I prefer seconds.
41     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
42
43     return root, uiApplication
44
45
46 def createTemplateScenario(root):
47     # IAgStkObjectRoot root: STK Object Model Root
48     root.NewScenario('Script.Scenario.Template')
49
50     # Switch units back temporarily
51     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('UTC')
52
53     startDatetime = '1 Jan 2019 12:00:00.000'
54     stopDatetime = '1 Jan 2020 12:00:00.000'
55     scen = root.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
56     scen.SetTimePeriod(startDatetime, stopDatetime)
57
58     # And reset the units before we return
59     root.UnitPreferences.Item('DateFormat').SetCurrentUnit('EpSec')
60
61
62 def sanitizeName(name):
63     # Function to remove prohibited characters from a satellite's name
64     badChars = [' ', '(', ')', '/', '\\']
65     rtnName = name
66     for c in badChars:
67         rtnName = rtnName.replace(c, '_')
68
69     return rtnName
70
71
72 def printTime(seconds):
73     m, s = divmod(seconds, 60)
74     h, m = divmod(m, 60)
75     return "%d Hours %02d Minutes %02d Seconds" % (h, m, s)
76
77
78 def getAssignedLongitude(sat):
79     # Grab the average longitude from the first day.
80     # 'Average' should make this work for non-geostationary as well
81     features = ['Detic.LLA.Longitude']
82     timeVals = [i for i in range(0, 60*60*24, 60)]
83     rtnArray = np.zeros(shape=(len(timeVals), len(features)))
84
85     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
86     info = dp.Item('Parameter Set: Cartographic Trajectory').QueryInterface(STKObjects.IAgDataProviderGroup)
```

```

87 group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
88 prov = group.Item('Trajectory(Cartographic)').QueryInterface(STKObjects.IAgDataPrvTimeVar)
89 results = prov.ExecSingleElementsArray(timeVals, features)
90
91 for i in range(len(features)):
92     rtnArray[:, i] = results.GetArray(i)
93
94 return np.mean(rtnArray[:, 0])
95
96
97 def getMetricData(sat, streakStart, streakEnd, features):
98     # Take a sample at the beginning and end of the streak
99     rtnArray = np.zeros(shape=(1, 2, len(features)))
100
101     # Pull spherical coordinates
102     dp = sat.DataProviders.QueryInterface(STKObjects.IAgDataProviderCollection)
103     info = dp.Item('Parameter Set: Orbit').QueryInterface(STKObjects.IAgDataProviderGroup)
104     group = info.Group.QueryInterface(STKObjects.IAgDataProviders)
105     prov = group.Item('Orbit').QueryInterface(STKObjects.IAgDataPrvTimeVar)
106     results = prov.ExecSingleElementsArray([streakStart, streakEnd], features)
107
108     for i in range(len(features)):
109         rtnArray[0, :, i] = results.GetArray(i)
110
111     return rtnArray
112
113
114 def addManeuver(targetSequence, dc, maneuverName):
115     dv = targetSequence.Segments.Insert(AgStkGatorLib.eVASegmentTypeManeuver, maneuverName, "-").QueryInterface(
116         AgStkGatorLib.IAgVAMCSManeuver)
117     dv.SetManeuverType(AgStkGatorLib.eVAManeuverTypeImpulsive)
118     impulsive = dv.Maneuver
119     impulsive.SetAttitudeControlType(AgStkGatorLib.eVAAttitudeControlThrustVector)
120
121     # Let the maneuver use all 3 dimensions
122     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianX)
123     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianY)
124     dv.EnableControlParameter(AgStkGatorLib.eVAControlManeuverImpulsiveCartesianZ)
125     xControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.X")
126     yControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Y")
127     zControlParam = dc.ControlParameters.GetControlByPaths(maneuverName, "ImpulsiveMnvr.Cartesian.Z")
128     xControlParam.Enable = True
129     yControlParam.Enable = True
130     zControlParam.Enable = True
131     xControlParam.MaxStep = 0.1
132     yControlParam.MaxStep = 0.1
133     zControlParam.MaxStep = 0.1
134
135     # Make it a bit easier to read
136     xControlParam.UseCustomDisplayUnit = True
137     yControlParam.UseCustomDisplayUnit = True
138     zControlParam.UseCustomDisplayUnit = True
139     xControlParam.CustomDisplayUnit = "m/sec"
140     yControlParam.CustomDisplayUnit = "m/sec"
141     zControlParam.CustomDisplayUnit = "m/sec"
142
143     dv = dv.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
144     return dv
145
146
147 def addPropagateByDuration(ts, propName, duration):
148     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
149         IAgVAMCSPropagate)
150     propagate.PropagatorName = "Earth Point Mass"
151     properties = propagate.StoppingConditions["Duration"].Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
152     properties.Trip = duration
153     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
154     return propagate
155
156
157 def addPropagateToAscendingNode(ts, propName, ascending=True):
158     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
159         IAgVAMCSPropagate)
160     propagate.PropagatorName = "Earth Point Mass"
161     if ascending:
162         properties = propagate.StoppingConditions.Add("AscendingNode").Properties.QueryInterface(AgStkGatorLib.
163             IAgVASToppingCondition)
164     else:
165         properties = propagate.StoppingConditions.Add("DescendingNode").Properties.QueryInterface(AgStkGatorLib.
166             IAgVASToppingCondition)
167     propagate.StoppingConditions.Remove("Duration")
168     properties.CoordSystem = "CentralBody/Earth Fixed"
169     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
170     return propagate
171
172
173 def addPropagateToApoapsis(ts, propName, apoapsis=True):
174     propagate = ts.Segments.Insert(AgStkGatorLib.eVASegmentTypePropagate, propName, "-").QueryInterface(AgStkGatorLib.
175         IAgVAMCSPropagate)
176     propagate.PropagatorName = "Earth Point Mass"
177     if apoapsis:
178         properties = propagate.StoppingConditions.Add("Apoapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
179     else:
180         properties = propagate.StoppingConditions.Add("Periapsis").Properties.QueryInterface(AgStkGatorLib.IAgVASToppingCondition)
181     propagate.StoppingConditions.Remove("Duration")
182     propagate = propagate.QueryInterface(AgStkGatorLib.IAgVAMCSSegment)
183     return propagate
184
185
186 def addNSCorrection(ts, dc):
187     dayLength = 60 * 60 * 24
188
189     # Propagate to ascending node
190     propName = "Propagate to Ascending Node"
191     addPropagateToAscendingNode(ts, propName, ascending=True)
192
193     # Maneuver

```

```

188 addManeuver(ts, dc, "Maneuver 1")
189
190 # Propagate 0.25 days - check latitude
191 propName = "Propagate for Quarter Day 1"
192 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
193 prop.Results.Add("Geodetic/Latitude")
194 r = dc.Results.GetResultByPaths(propName, "Latitude")
195 r.Enable = True
196 r.DesiredValue = 0.0
197 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
198
199 # Propagate to descending node
200 propName = "Propagate to Descending Node"
201 addPropagateToAscendingNode(ts, propName, ascending=False)
202
203 # Maneuver
204 addManeuver(ts, dc, "Maneuver 2")
205
206 # Propagate 0.25 days - check latitude
207 propName = "Propagate for Quarter Day 2"
208 prop = addPropagateByDuration(ts, propName, 0.25 * dayLength)
209 prop.Results.Add("Geodetic/Latitude")
210 r = dc.Results.GetResultByPaths(propName, "Latitude")
211 r.Enable = True
212 r.DesiredValue = 0.0
213 r.Tolerance = 0.05 # 0.1 deg box, so 0.05 in either direction
214
215
216 def addEWTransfer(ts, dc, assignedLong, longTolerance=0.05):
217     # Propagate to Apoapsis
218     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
219
220     # Maneuver (no checks)
221     addManeuver(ts, dc, "Maneuver 1")
222
223     # Propagate to Periapasis (shouldn't be needed, just giving another maneuver)
224     addPropagateToApoapsis(ts, "Propagate to Periapasis 1", apoapsis=False)
225
226     # Maneuver (no checks)
227     addManeuver(ts, dc, "Maneuver 2")
228
229     # Propagate to Apoapsis
230     addPropagateToApoapsis(ts, "Propagate to Apoapsis 1", apoapsis=True)
231
232     # Maneuver - check periapsis
233     maneuverName = "Maneuver 3"
234     maneuver = addManeuver(ts, dc, maneuverName)
235     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Periapsis")
236     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Periapsis")
237     r.Enable = True
238     r.DesiredValue = 35786.0
239     r.Tolerance = 1.0
240
241     # Propagate to Periapsis
242     addPropagateToApoapsis(ts, "Propagate to Periapsis 2", apoapsis=False)
243
244     # Maneuver - check apoapsis, eccentricity, longitude
245     maneuverName = "Maneuver 4"
246     maneuver = addManeuver(ts, dc, maneuverName)
247     maneuver.Results.Add("Keplerian Elems/Altitude.Of.Apoapsis")
248     r = dc.Results.GetResultByPaths(maneuverName, "Altitude.Of.Apoapsis")
249     r.Enable = True
250     r.DesiredValue = 35786.0
251     r.Tolerance = 1.0
252     maneuver.Results.Add("Keplerian Elems/Eccentricity")
253     r = dc.Results.GetResultByPaths(maneuverName, "Eccentricity")
254     r.Enable = True
255     r.DesiredValue = 0.0
256     r.Tolerance = 0.001
257     maneuver.Results.Add("Geodetic/Longitude")
258     r = dc.Results.GetResultByPaths(maneuverName, "Longitude")
259     r.Enable = True
260     r.DesiredValue = assignedLong
261     r.Tolerance = longTolerance
262
263
264 def maneuverSchedule(satDriver, stopTime, longSchedule):
265     # Suppose that each GEO slot is a 0.1 X 0.1 deg square.
266     dayLength = 60 * 60 * 24
267     time = 0
268     dayNum = 0
269     schedulePos = -1
270     assignedLong = 0
271     count = 0
272     while time < stopTime:
273         # Figure out where we are in the schedule
274         if dayNum % 31 == 0:
275             schedulePos += 1
276
277         # Repeat the schedule if time remains
278         prevLong = assignedLong
279         assignedLong = longSchedule[schedulePos % len(longSchedule)]
280         accuracy = 10
281
282         # Setup the target sequence and differential corrector
283         ts = satDriver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeTargetSequence, "Day " +
284             str(dayNum) + ": Maneuver Schedule", "-").QueryInterface(AgStkGatorLib.IAgVAMCSTargetSequence)
285         dc = ts.Profiles.Item(0).QueryInterface(AgStkGatorLib.IAgVAProfileDifferentialCorrector)
286         dc.EnableDisplayStatus = False
287         dc.ClearCorrectionsBeforeRun = True
288         dc.EnableBPlaneNominal = True
289         dc.EnableBPlanePerturbations = True
290         dc.EnableHomotopy = True # Seems to help find solutions progressively.
291         dc.HomotopySteps = 1000
292         dc.RootFindingAlgorithm = 1 # The Newton-Raphson method seems to be more accurate than Secant, but slower
293         #dc.DerivativeCalcMethod = 1 # Use the central difference instead of a forward difference
294         dc.Mode = AgStkGatorLib.eVAProfileModelIterate
295         dc.MaxIterations = 200 # Iterations seems to make little difference once Homotopy fails.

```

```

295     if count % 2 == 0:
296         addNSCorrection(ts, dc)
297         numDays = 1
298     else:
299         print("On Day " + str(dayNum) + ": transferring from " + str(prevLong) + " long to " +
300               str(assignedLong) + " long")
301         addEWTransfer(ts, dc, assignedLong, longTolerance=accuracy)
302         numDays = 2
303
304         # Progressively require a stricter tolerance on your station
305         if accuracy == 10:
306             accuracy = 5
307         elif accuracy == 5:
308             accuracy = 1
309         elif accuracy == 1:
310             accuracy = 0.05
311
312     # Run the MCS with this new day added
313     ts.Action = AgStkGatorLib.eVATargetSeqActionRunActiveProfiles
314     print("Starting MCS for day " + str(dayNum))
315     errorCode = satDriver.RunMCS2()
316     if errorCode != 0:
317         print("MCS Error")
318         #satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".broken")
319     else:
320         print("MCS Successful")
321         # Save the scenario in case debugging is needed
322         #stkRoot.SaveScenarioAs(os.path.join(stkLocation, 'astrogator_debug_scenario'))
323         #satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".e")
324
325     # Done with this day
326     count += 1
327     dayNum += numDays
328     time += dayLength * numDays
329
330
331 def addTLEWorker(name, ssc, filepath, ephemerisPath, stkRoot):
332     print("Starting satellite: " + str(name))
333     count = 0
334     jobStartTime = time.time()
335     outFilePath = os.path.join(ephemerisPath, name)
336
337     # Dump the old file (if one exists)
338     if os.path.exists(outFilePath + ".e"):
339         os.remove(outFilePath + ".e")
340     if os.path.exists(outFilePath + ".broken"):
341         os.remove(outFilePath + ".broken")
342
343     # Do the work for this satellite
344     try:
345         # Have to build the schedule before anything is loaded
346         longSchedule = createLongitudeSchedule(tleFile, stkRoot, name)
347
348         # Ready to load this element into the scenario
349         satellite = stkRoot.CurrentScenario.Children.New(18, name)
350         satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
351
352         satellite2.SetPropagatorType(4) # ePropagatorSGP4
353         propagator = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorSGP4)
354         commonTasks = propagator.CommonTasks.QueryInterface(STKObjects.IAgVePropagatorSGP4CommonTasks)
355         commonTasks.AddSegsFromFile(ssc, filepath)
356         propagator.Propagate()
357
358         # Now pull the ephemeris at scenario start and switch to astrogator
359         x = getMetricData(satellite, 0, 0,
360                           ["Classical.SemimajorAxis", "Classical.Eccentricity", "Classical.Inclination",
361                            "Classical.RAAN", "Classical.ArgumentOfPeriapsis", "Classical.MeanAnomaly"])
362
363         satellite2.SetPropagatorType(12) # Type 12 is Astrogator
364         driver = satellite2.Propagator.QueryInterface(AgStkGatorLib.IAgVADriverMCS)
365         driver.MainSequence.RemoveAll()
366
367         # Only run target sequences that have changed
368         options = driver.options
369         options.SmartRunMode = 1
370
371         # Initial State
372         initState = driver.MainSequence.Insert(AgStkGatorLib.eVASegmentTypeInitialState, "Initial State",
373                                                "12").QueryInterface(AgStkGatorLib.IAgVAMCSInitialState)
374         initState.SetElementType(AgStkGatorLib.eVAElementTypeKeplerian)
375         initState.OrbitEpoch = 0 # Scenario start time
376         kep = initState.Element.QueryInterface(AgStkGatorLib.IAgVAElementKeplerian)
377
378         # For some reason these come back out of order...
379         # Moments like this make me hate STK quite deeply
380         kep.SemiMajorAxis = x[0, 0, 4]
381         kep.Eccentricity = x[0, 0, 0]
382         kep.Inclination = x[0, 0, 1]
383         kep.RAAN = x[0, 0, 2]
384         kep.ArgOfPeriapsis = x[0, 0, 3]
385         kep.MeanAnomaly = x[0, 0, 5]
386
387         scen = stkRoot.CurrentScenario.QueryInterface(STKObjects.IAgScenario)
388         driver.Options.ClearDWCGraphicsBeforeEachRun = True
389         maneuverSchedule(driver, scen.StopTime, longSchedule)
390
391         # Done, so save
392         satellite2.ExportTools.GetEphemerisStkExportTool().Export(outFilePath + ".e")
393
394         count += 1
395         timeElapsed = time.time() - jobStartTime
396
397     finally:
398         print("Completed satellite: " + str(name))
399         print("Time Elapsed: " + printTime(timeElapsed))
400
401

```

```

402 def addTLEFile(filepath, ephemerisPath, satName, stkRoot):
403
404     f = open(filepath, 'r')
405     filelines = f.readlines()
406     f.close()
407
408     # In my case, every line of the TLE starts with the line number
409     # I have seen examples where line 0 doesn't, but hope that doesn't
410     # happen in our set.
411     for line in filelines:
412         if line.startswith('0'):
413             name = sanitizeName(line[2:-1])
414         elif line.startswith('1'):
415             # Because someone thought it would be a good idea to stick classification on the end with no whitespace...
416             number = line.split()[1]
417             ssc = number[:-1]
418         elif line.startswith('2') and (float(line.split()[2]) < 1.0):
419             if name == satName:
420                 addTLEWorker(name, ssc, filepath, ephemerisPath, stkRoot)
421
422
423 def createLongitudeSchedule(filepath, stkRoot, startSatName):
424     # Start by pulling a list of the satellites we care about
425     namesToSearch = ["JCSAT-16", "INTELSAT-907", "GOES-13", "AMC-16", "GOES-16", "DIRECTV-11", "MEXSAT-3", "GALAXY-15"]
426
427     # Read the file in
428     f = open(filepath, 'r')
429     filelines = f.readlines()
430     f.close()
431
432     schedule = []
433     startIndex = None
434     for line in filelines:
435         if line.startswith('0'):
436             name = sanitizeName(line[2:-1])
437         elif line.startswith('1'):
438             # Because someone thought it would be a good idea to stick classification on the end with no whitespace...
439             number = line.split()[1]
440             ssc = number[:-1]
441         elif line.startswith('2') and (float(line.split()[2]) < 1.0):
442             if name in namesToSearch:
443                 if name == startSatName:
444                     startIndex = len(schedule)
445
446             # Ready to load this element into the scenario
447             satellite = stkRoot.CurrentScenario.Children.New(18, name)
448             satellite2 = satellite.QueryInterface(STKObjects.IAgSatellite)
449
450             satellite2.SetPropagatorType(4) # ePropagatorSGP4
451             propagator = satellite2.Propagator.QueryInterface(STKObjects.IAgVePropagatorSGP4)
452             commonTasks = propagator.CommonTasks.QueryInterface(STKObjects.IAgVePropagatorSGP4CommonTasks)
453             commonTasks.AddSegsFromFile(ssc, filepath)
454             propagator.Propagate()
455
456             # Get this satellite's longitude slot
457             longitudeSlot = getAssignedLongitude(satellite)
458             schedule.append(longitudeSlot)
459             satellite.Unload()
460
461     # Now sort so that we start at the desired satellite
462     return schedule[startIndex:] + schedule[:startIndex]
463
464
465 if __name__ == "__main__":
466     print("Starting Script...")
467     tleFile = os.path.abspath('geo-catalog-31july2017.tle')
468     satList = ["JCSAT-16", "INTELSAT-907", "GOES-13", "AMC-16", "GOES-16", "DIRECTV-11", "GALAXY-15"]
469     for satName in satList:
470         try:
471             # Start new STK instance
472             stkRoot, uiApp = startSTK(False)
473             createTemplateScenario(stkRoot)
474
475             addTLEFile(os.path.abspath(tleFile), os.path.abspath('Ephemeris.Maneuvering'), satName, stkRoot)
476         finally:
477             stkRoot.CloseScenario()
478             del stkRoot
479             del uiApp
480     print("Script Complete. Exiting...")

```

## H.2 Experiment Launcher

Listing H.2: Cluster Experiment Launcher for Maneuvering Experiment

```

1 # Title: Launcher for Maneuvering Experiment
2 # Author: IstLt Ian McQuaid
3 # Organization: Air Force Institute of Technology
4 # Date: 3 August 2018
5 # Description: Launches the maneuvering experiment on the HPC. This uses the cluster experiment framework, along
6 # with the keras experiment ANN code to run many machine learning models over a short period of time. This experiment
7 # examines all pairs of RSOs.
8
9 #!/usr/bin/python
10 # Example PBS cluster job submission in Python
11
12 import argparse
13 import os
14 import tensorflow as tf
15

```

```

16 # import pyhpc as hpc
17 # hpc.ClusterExperiment()
18
19 from cluster_experiment import ClusterExperiment
20
21
22 def main(FLAGS):
23
24     # Clear and remake the log directory.
25     if tf.gfile.Exists(FLAGS.log_dir):
26
27         tf.gfile.DeleteRecursively(FLAGS.log_dir)
28
29     tf.gfile.MakeDirs(FLAGS.log_dir)
30
31     # Instantiate an experiment.
32     exp = ClusterExperiment()
33
34     # Set the number of reps for each config.
35     exp.set_rep_count(5)
36
37     # Set independent parameters.
38     exp.add.design('batch_size', [32])
39     exp.add.design('max_epochs', [10000000000])
40     exp.add.design('learning_rate', [0.00001])
41     exp.add.design('dropout_rate', [0.0])
42     exp.add.design('patience', [30])
43     exp.add.design('training_set', ["Maneuvering_Dataset"])
44     exp.add.design('class_a', ['Satellite_1', 'Satellite_2', 'Satellite_3', 'Satellite_4', 'Satellite_5', 'Satellite_6',
45                                'Satellite_7', 'Satellite_8'])
46     exp.add.design('class_b', ['Satellite_1', 'Satellite_2', 'Satellite_3', 'Satellite_4', 'Satellite_5', 'Satellite_6',
47                                'Satellite_7', 'Satellite_8'])
48
49     exp.add.design('model_type', ["photo_convnet"])
50
51     # Launch the experiment.
52     exp.launch_experiment(exp_filename=FLAGS.experiment_py_file,
53                          log_dir=FLAGS.log_dir,
54                          account_str='MHPCC96670DAI',
55                          queue_str='standard',
56                          module_str='anaconda2/5.0.1 gcc/5.3.0 cudnn/6.0 tensorflow/1.8.0',
57                          manager='pbs',
58                          shuffle_job_order=True)
59
60     # Manually note response variables.
61     response_labels = ['train_acc_1',
62                       'test_acc_1',
63                       'train_acc_2',
64                       'test_acc_2',
65                       'train_acc_3',
66                       'test_acc_3',
67                       'num_epochs',
68                       'total_runtime',
69                       'time_per_epoch',]
70
71     # Wait for the output to return.
72     exp.join_job_output(FLAGS.log_dir,
73                        FLAGS.log_filename,
74                        FLAGS.max_runtime,
75                        response_labels)
76
77     print("All jobs complete. Exiting.")
78
79
80 if __name__ == '__main__':
81
82     # Instantiate an arg parser.
83     parser = argparse.ArgumentParser()
84
85     parser.add_argument('--log_dir', type=str,
86                        default=os.path.join(os.environ["WORKDIR"], 'log'),
87                        help='Summaries log directory.')
88
89     parser.add_argument('--log_filename', type=str,
90                        default='experiment_results.csv',
91                        help='Merged output filename.')
92
93     parser.add_argument('--max_runtime', type=int,
94                        default=3600000,
95                        help='Number of seconds to run before giving up.')
96
97     parser.add_argument('--experiment_py_file', type=str,
98                        default=os.path.join(os.environ["WORKDIR"], 'experiment_photo_pairwise_stationkeep', '
99                                             keras_experiment.py'),
100                        help='Name of the experiment file.')
101
102     # Parse known arguments.
103     FLAGS, unparsed = parser.parse_known_args()
104     main(FLAGS)

```

## Bibliography

- [1] “DJILP — Denver Journal of International Law and Policy”. URL <http://djilp.org/3494/the-bogota-declaration-and-the-curious-case-of-geostationary-orbit/>.
- [2] “HPC Centers - Non-Allocated Systems”. URL <https://centers.hpc.mil/systems/nonallocated.html{#}Hokulea>.
- [3] “MIT Lincoln Laboratory: Advanced Imager Technology: Space Surveillance Telescope”. URL <https://www.ll.mit.edu/mission/electronics/ait/scientific-dod-imagers/space-surveillance-telescope.html>.
- [4] “Space Based Space Surveillance Fact Sheet”. URL <http://www.afspc.af.mil/About-Us/Fact-Sheets/Article/249017/space-based-space-surveillance-sbss/>.
- [5] “Space Surveillance Sensors: GEODSS (Ground-based Electro-Optical Deep Space Surveillance) System (August 20, 2012) — mostlymissiledefense”. URL <https://mostlymissiledefense.com/2012/08/20/space-surveillance-sensors-geodss-ground-based-electro-optical-deep-space-surveillance-system-august-20-2012/>.
- [6] “Space-Track.Org”. <https://www.space-track.org/>. Accessed: 2017-07-31.
- [7] Abdel-Hamid, O, A Mohamed, H Jiang Acoustics, Speech And, and undefined 2012. “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition”. *researchgate.net*. URL <https://www.researchgate.net/profile/Ossama{ }Abdel-Hamid/publication/261119155{ }Applying{ }Convolutional{ }Neural{ }Networks{ }concepts{ }to{ }hybrid{ }NN-HMM{ }model{ }for{ }speech{ }recognition/links/559407fd08ae5af2b0ecf4fb.pdf>.
- [8] Aphex34. “Typical cnn”, 2015. URL [https://commons.wikimedia.org/wiki/File:Typical\\_cnn.png](https://commons.wikimedia.org/wiki/File:Typical_cnn.png).
- [9] Bass, Michael, Casimer DeCusatis, Jay Enoch, Vasudevan Lakshminarayanan, Guifang Li, Carolyn Macdonald, Virendra Mahajan, and Eric Van Stryland. *Handbook of Optics, Third Edition Volume II: Design, Fabrication and Testing, Sources and Detectors, Radiometry and Photometry*. McGraw-Hill, Inc., New York, NY, USA, 3 edition, 2010. ISBN 0071498907, 9780071498906.
- [10] Bate, Roger., Donald D. Mueller, and Jerry E. White. *Fundamentals of astrodynamics*. Dover Publications, 1971. ISBN 0486600610. URL <https://books.google.com/books/about/Fundamentals{ }of{ }Astrodynamics.html?id=UtJK8cetqGkC>.
- [11] Chen, Yanping, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. “The UCR Time Series Classification Archive”, July 2015. [www.cs.ucr.edu/~eamonn/time\\_series\\_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/).

- [12] Chollet, François et al. “Keras”. <https://keras.io>, 2015.
- [13] Chu, Donald, Sam Chen, Derrick Early, Doug Freesland, Alexander Krimchansky, Bo Naasz, Alan Reth, Kumar Tadikonda, John Tsui, and Tim Walsh. “GOES-R STATIONKEEPING AND MOMENTUM MANAGEMENT”. jan 2006. URL <https://ntrs.nasa.gov/search.jsp?R=20060012315>.
- [14] Cui, Zhicheng, Wenlin Chen, and Yixin Chen. “Multi-Scale Convolutional Neural Networks for Time Series Classification”. mar 2016. URL <http://arxiv.org/abs/1603.06995>.
- [15] en>User:Cburnett. “Artificial Neural Network”, 2006. URL [https://commons.wikimedia.org/wiki/File:Artificial\\_neural\\_network.svg](https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg).
- [16] Fujimoto, Kohei, Daniel J. Scheeres, Johannes Herzog, and Thomas Schildknecht. “Association of optical tracklets from a geosynchronous belt survey via the direct Bayesian admissible region approach”. *Advances in Space Research*, 53(2):295–308, jan 2014. ISSN 0273-1177. URL <https://www.sciencedirect.com/science/article/pii/S0273117713007059>.
- [17] Gaylor, DE, BW Barbee AAS Meeting, and undefined 2007. “Algorithms for safe spacecraft proximity operations”. *academia.edu*. URL <http://www.academia.edu/download/32360612/AAS{ }2007{ }107.pdf>.
- [18] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [19] Greve, Gabriel H, Kenneth M Hopkinson, and Gary B Lamont. “Evolutionary Sensor Allocation for the Space Surveillance Network”. 1–26, 2016.
- [20] Hall, D, B Calef, K Knox, and M Bolden. “Separating attitude and shape effects for non-resolved objects”. *The 2007 AMOS*, 2007. URL <http://amostech.com/TechnicalPapers/2007/NROC/Hall.pdf>.
- [21] Hefron, Ryan, Brett Borghetti, Christine Schubert Kabban, James Christensen, and Justin Estepp. “Cross-participant EEG-based assessment of cognitive workload using multi-path convolutional recurrent neural networks”. *Sensors (Switzerland)*, 18(5), apr 2018. ISSN 14248220. URL <http://www.ncbi.nlm.nih.gov/pubmed/29701668http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5982227>.
- [22] Hill, K, P Sydney, K Hamada, R Cortez Paper AAS 10-150 ..., and undefined 2010. “Covariance-based network tasking of optical sensors”. *researchgate.net*. URL <https://www.researchgate.net/profile/Moriba{ }Jah/publication/265987703{ }COVARIANCE-BASED{ }NETWORK{ }TASKING{ }OF{ }OPTICAL{ }SENSORS/links/55a3e79208aef604aa03c3ad/COVARIANCE-BASED-NETWORK-TASKING-OF-OPTICAL-SENSORS.pdf>.

- [23] Hill, Keric, Paul Sydney, Randy Cortez, Kris Hamada, Daron Nishimoto, Kim Luu, and Paul W Schumacher, Jr. “Dynamic Tasking of Networked Sensors Using Covariance Information”, 2010. URL <http://www.dtic.mil/docs/citations/ADA531704>.
- [24] Hobson, TA, , I Clarkson 2011 Advanced Maui Optical Space, and undefined 2011. “Sensor-scheduling simulation of disparate sensors for Space Situational Awareness”. *amostech.com*. URL <https://www.amostech.com/TechnicalPapers/2011/Poster/HOBSON.pdf>.
- [25] Hussein, II, CW Roscoe, and MP Wilkins. “Information theoretic criteria for observation-to-observation association”. 2014. URL <http://www.dtic.mil/docs/citations/ADA616791>.
- [26] Hussein, II, CWT Roscoe, MP Wilkins Proceedings of the . . . , and undefined 2015. “Track-to-Track Association Using Bhattacharyya Divergence”. *amostech.com*. URL <https://www.amostech.com/TechnicalPapers/2015/Astrodynamics/Hussein.pdf>.
- [27] Jeas, W. C. and R. Anctil. “The Ground-based Electro-Optical Deep Space Surveillance /GEODSS/ system”. *Military Electronics/Countermeasures*, vol. 7, Nov. 1981, p. 47-51., 7:47–51, 1981. URL <http://adsabs.harvard.edu/abs/1981MiElC...7...47J>.
- [28] Kaasalainen, M and J Torppa. “Optimization Methods for Asteroid Lightcurve Inversion I. Shape Determination”. *Icarus*, 153:24–36, 2001. URL <http://www.idealibrary.com>.
- [29] Kennewell, JA, BN Vo FUSION, and undefined 2013. “An overview of space situational awareness.” *ba-ngu.vo-au.com*. URL <http://ba-ngu.vo-au.com/vo/KV{ }SSA{ }FUSION13.pdf>.
- [30] Kingma, Diederik P. and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. dec 2014. URL <http://arxiv.org/abs/1412.6980>.
- [31] Krizhevsky, A, I Sutskever, GE Hinton Advances in Neural, and undefined 2012. “Imagenet classification with deep convolutional neural networks”. *papers.nips.cc*. URL <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>.
- [32] Kurosaki, Hirohisa. “Observation of Light Curves of Space Objects”. 2009.
- [33] Leafnode. “Hohmann transfer orbit”, 2007. URL [https://commons.wikimedia.org/wiki/File:Hohmann\\_transfer\\_orbit.svg](https://commons.wikimedia.org/wiki/File:Hohmann_transfer_orbit.svg).
- [34] Mark Ackermann, Rex Kiziah, Peter Zimmer, John McGraw, David Cox. “A SYSTEMATIC EXAMINATION OF GROUND-BASED AND SPACE-BASED APPROACHES TO OPTICAL DETECTION AND TRACKING OF SATELLITES”. *31st Space Symposium*, 2015.

- [35] Maruskin, JM, DJ Scheeres, and KT Alfriend. “Correlation of optical observations of objects in earth orbit”. *Journal of Guidance, Control,*, 2009. URL <https://arc.aiaa.org/doi/pdfplus/10.2514/1.36398>.
- [36] McCall, Gene H and John H Darrah. “Space Situational Awareness: Difficult, Expensive and Necessary”, 2014. URL <http://www.dtic.mil/docs/citations/ADA625656>.
- [37] McClain, W.D. and D.A. Vallado. *Fundamentals of Astrodynamics and Applications*. Space Technology Library. Springer Netherlands, 2001. ISBN 9780792369035. URL <https://books.google.com/books?id=PJLIWzMBKjkC>.
- [38] Milani, A, GF Gronchi, MM Vitturi, and Z Knežević. “Orbit determination with very short arcs. I admissible regions”. *Celestial Mechanics and*, 2004. URL <http://www.springerlink.com/index/k1p64x4197581046.pdf>.
- [39] Miller, James G. (Gil). “A New Sensor Allocation Algorithm for the Space Surveillance Network”, 2007. URL <https://www.jstor.org/stable/43941064>.
- [40] Newman, Andrew J., Sean R. Martin, Benjamin M. Rodriguez, Nishant L. Mehta, and Eric M. Klatt. “A comparison of stochastic optimizers applied to dynamic sensor scheduling for satellite tracking”. volume 7697, 76970M. International Society for Optics and Photonics, apr 2010. URL <http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.850204>.
- [41] Ostro, SJ, R Connelly, and M Dorogi. “Convex-profile inversion of asteroid lightcurves: Theory and applications”. *Icarus*, 1988. URL <http://www.sciencedirect.com/science/article/pii/0019103588901261>.
- [42] Petrick, Beth L. “Weighting Scheme for the Space Surveillance Network Automated Tasker.”, 1994. URL <http://www.dtic.mil/docs/citations/ADA289285>.
- [43] Rodriquez, HM, KJ Abercromby, KS Jarvis, and E Barker. “Using Light Curves to Characterize Size and Shape of Pseudo-Debris”. 2006. URL <https://ntrs.nasa.gov/search.jsp?R=20060024592>.
- [44] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [45] Schulman, Eric and Caroline V. Cox. “Misconceptions about astronomical magnitudes”. *American Journal of Physics*, 65(10):1003–1007, oct 1997. ISSN 0002-9505. URL <http://aapt.scitation.org/doi/10.1119/1.18714>.
- [46] Scott, A., John Hackett, and K. Man. “On-Orbit Results for Canada’s Sapphire Optical Payload”. *Amotech*, 2013. URL <https://amotech.com/TechnicalPapers/2013/Space-Based{ }Assets/SCOTT.pdf>.

- [47] Siminski, JA, H Fiedler, T Schildknecht Proceedings of the 6th, and undefined 2013. “Track association performance of the best hypotheses search method”. *aiuws.unibe.ch*. URL <http://aiuws.unibe.ch/ccd/publist/data/2013/artproc/TS{ }ESDC2013a.pdf>.
- [48] Siminski, JA, O Montenbruck, H Fiedler Advances in the . . . , and undefined 2013. “Best Hypothesis Search on Iso-Energy-Grid for Initial Orbit Determination and Track Association”. *core.ac.uk*. URL <https://core.ac.uk/download/pdf/18591938.pdf>.
- [49] Siminski, J.A., O. Montenbruck, H. Fiedler, and T. Schildknecht. “Short-arc tracklet association for geostationary objects”. *Advances in Space Research*, 53(8):1184–1194, apr 2014. ISSN 0273-1177. URL <https://www.sciencedirect.com/science/article/pii/S0273117714000520>.
- [50] Singh, N, JT Horwood, JM Aristoff, AB Poore, and C Sheaff. “Multiple hypothesis tracking (MHT) for space surveillance: Results and simulation studies”. 2013. URL <http://www.dtic.mil/docs/citations/ADA591392>.
- [51] Stauch, J, M Jah, J Baldwin, T Kelecyc AIAA/AAS Astrodynamics . . . , and undefined 2014. “Mutual application of joint probabilistic data association, filtering, and smoothing techniques for robust multiple space object tracking”. *arc.aiaa.org*. URL <https://arc.aiaa.org/doi/pdf/10.2514/6.2014-4365>.
- [52] Sugiyama, Masashi. *Introduction to Statistical Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016. ISBN 9780128023501.
- [53] Taff, L G. “ON INITIAL ORBIT DETERMINATION”. *THE ASTRONOMICAL JOURNAL*, 89(9), 1984. URL <http://adsbit.harvard.edu/cgi-bin/nph-iarticle{ }query?1984AJ.....89.1426T{ }defaultprint=YES{ }filetype=.pdf>.
- [54] Tfr000. “Ra and dec on celestial sphere”, 2012. URL [https://commons.wikimedia.org/wiki/File:Ra\\_and\\_dec\\_on\\_celestial\\_sphere.png](https://commons.wikimedia.org/wiki/File:Ra_and_dec_on_celestial_sphere.png).
- [55] Tommei, G, A Milani, and A Rossi. “Orbit determination of space debris: admissible regions”. *Celestial Mechanics and Dynamical*, 2007. URL <http://www.springerlink.com/index/45880W867P5JQ862.pdf>.
- [56] W Viehmann, R E Predmore. “Ultraviolet And Visible Brdf Data On Spacecraft Thermal Control And Optical Baffle Materials”, 1987. URL <https://doi.org/10.1117/12.939484>.
- [57] Wang, Z, T Oates Workshops at the Twenty-Ninth AAAI Conference On, and undefined 2015. “Encoding time series as images for visual inspection and classification using tiled convolutional neural networks”. *aaai.org*. URL <http://www.aaai.org/ocs/index.php/WS/AAAIW15/paper/viewFile/10179/10251>.

- [58] Wang, Z, W Yan, T Oates Neural Networks (IJCNN), undefined 2017, and undefined 2017. “Time series classification from scratch with deep neural networks: A strong baseline”. *ieeexplore.ieee.org*. URL <https://ieeexplore.ieee.org/abstract/document/7966039/>.
- [59] Weber, Robert. “The Ground-Based Electro-Optical Detection Of Deep-Space Satellites”. volume 0143, 59–69. International Society for Optics and Photonics, sep 1978. URL <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1227351>.
- [60] Woffinden, 1976, David C. “On-orbit satellite inspection : navigation and [Delta]v analysis”. 2004. URL <https://dspace.mit.edu/handle/1721.1/28862>.
- [61] Yang, J, MN Nguyen, PP San, X Li, S Krishnaswamy Ijcai, and undefined 2015. “Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.” *aaai.org*. URL <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/download/10710/11297>.
- [62] Zheng, Yi, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. “Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks”. 298–310. 2014. URL [http://link.springer.com/10.1007/978-3-319-08010-9\\_{\\_}33](http://link.springer.com/10.1007/978-3-319-08010-9_{_}33).

| <b>REPORT DOCUMENTATION PAGE</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                             |                                          |                                   |                                   | <i>Form Approved</i><br><i>OMB No. 0704-0188</i>                                                  |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|------------------------------------------|-----------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------|--|
| 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 this 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 any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. <b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>1. REPORT DATE</b> (DD-MM-YYYY)<br>01-09-2018                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                             | <b>2. REPORT TYPE</b><br>Master's Thesis |                                   |                                   | <b>3. DATES COVERED</b> (From — To)<br>Sep 2016-Sep 2018                                          |  |
| <b>4. TITLE AND SUBTITLE</b><br><br>Autonomous Association of GEO RSO Observations using Deep Neural Networks                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                             |                                          |                                   | <b>5a. CONTRACT NUMBER</b>        |                                                                                                   |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   | <b>5b. GRANT NUMBER</b>           |                                                                                                   |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   | <b>5c. PROGRAM ELEMENT NUMBER</b> |                                                                                                   |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   | <b>5d. PROJECT NUMBER</b>         |                                                                                                   |  |
| <b>6. AUTHOR(S)</b><br><br>McQuaid, Ian W., First Lieutenant, USAF                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                             |                                          |                                   | <b>5e. TASK NUMBER</b>            |                                                                                                   |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   | <b>5f. WORK UNIT NUMBER</b>       |                                                                                                   |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b><br>Air Force Institute of Technology<br>Graduate School of Engineering and Management (AFIT/EN)<br>2950 Hobson Way<br>WPAFB, OH 45433-7765                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                                          |                                   |                                   | <b>8. PERFORMING ORGANIZATION REPORT NUMBER</b><br><br>AFIT-ENG-MS-18-S-014                       |  |
| <b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                             |                                          |                                   |                                   | <b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>                                                           |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   |                                   | <b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>                                                     |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b><br>DISTRIBUTION STATEMENT A:<br>APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>13. SUPPLEMENTARY NOTES</b><br>This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>14. ABSTRACT</b><br>Ground-based non-resolved optical observations of resident space objects (RSOs) in geosynchronous orbit (GEO) represent the majority of the space surveillance network's (SSN's) deep-space tracking. Reliable and accurate tracking necessitates temporal separation of the observations. This requires that subsequent observations be associated with prior observations of a given RSO before they can be used to create or refine that RSO's ephemeris. The use of astrometric data (e.g. topocentric angular position) alone for this association task is complicated by RSO maneuvers between observations, and by RSOs operating in close proximity. Accurately associating an observation with an RSO thus motivates the use of photometric light curves in that association process. Contemporary machine learning, specifically deep neural networks (DNNs), offers mechanisms to perform that association autonomously by first learning patterns between observations and their parent RSOs. This research assesses the extent to which a trained DNN can autonomously associate previously unseen observations with the RSO they represent. DNN performance is assessed by recording the percentage of observations from a held-out testing set associated to the correct RSO. The performance of the DNN is evaluated in several simulated scenarios. In the first, the RSOs maintain station for one year. In the second, the RSOs station keep in progressively closer proximity. In the third, the RSOs permute stations with one another. In each case the photometric observations are generated as if observed by the Maui GEODSS site. This research contributes to the development of autonomous SSN telescopes, systems to distinguish closely spaced RSOs, and systems to autonomously and rapidly update an RSO's ephemeris after it maneuvers. |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>15. SUBJECT TERMS</b><br>space situational awareness, track association, observation association, deep neural networks, deep learning, deep convolutional neural networks, UCTs, UCOs                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                             |                                          |                                   |                                   |                                                                                                   |  |
| <b>16. SECURITY CLASSIFICATION OF:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                             |                                          | <b>17. LIMITATION OF ABSTRACT</b> |                                   | <b>18. NUMBER OF PAGES</b>                                                                        |  |
| <b>a. REPORT</b><br><br>U                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <b>b. ABSTRACT</b><br><br>U | <b>c. THIS PAGE</b><br><br>U             | <br><br>UU                        |                                   | <br><br>119                                                                                       |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   |                                   | <b>19a. NAME OF RESPONSIBLE PERSON</b><br>Dr. Laurence D. Merkle (ENG)                            |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                             |                                          |                                   |                                   | <b>19b. TELEPHONE NUMBER</b> (include area code)<br>(937) 255-3636 x4526 Laurence.Merkle@afit.edu |  |