This page left intentionally blank
AN IMPROVED SCHEDULING ALGORITHM FOR SPACE-BASED SPACE SURVEILLANCE SENSORS

by

Aaron R. Newman

Submitted to the Department of Aeronautics and Astronautics on May 14, 1993 in partial fulfillment of the requirements for the Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

This thesis discusses the development of an N-step look ahead algorithm designed to efficiently schedule searches for resident space objects conducted by a space-based, angles-only optical sensor. The implementation of this algorithm is called the N-Step Ahead Single Object Scheduler, or N-STASOS.

Scheduling techniques currently employed by space surveillance sensors are presented and compared with the N-step look ahead strategy. In addition, a global scheduling strategy, and the application of optimal search theory to the problem are discussed.

The programming methodology and implementation of N-STASOS are presented, as are results of testing which took place at the Experiment Test System (ETS). ETS is a ground-based optical sensor site in Socorro, New Mexico run by the Massachusetts Institute of Technology's Lincoln Laboratory.

Thesis Advisor: Prof. Walter M. Hollister
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Dr. Grant H. Stokes
Title: Staff, MIT Lincoln Laboratory
Acknowledgements*

This paper is the culmination of over two years of work at MIT Lincoln Laboratory. I am grateful for the experience I gained at the lab and for the friends I made there. Now to the specifics... I must first thank Grant Stokes, who agreed to let a poor UROP student stay on for a Master's degree, and has been a constant source of support, even when constantly traveling on lab business. Thanks must also go to Gerry Banner and Mark Czerwinski, for providing support, both moral and financial. Kent Pollock was gracious enough to put up with my constant ORBLIB pestering, and Betsy Rozier, whose technical support and testing assistance was a great help, also introduced me to Mike Mulligan and Mary Ann. Susan Andrews and Ed Powell prepared test cases for me to take to ETS, so that I could concentrate on other things (like writing this paper).

Testing for this project would not have been possible without the assistance of all the staff at ETS in Socorro, New Mexico. Thanks in particular to Matthew Blythe, Eric Pearce, Peter Trujillo, and Debbie Borden for all their help.

I also owe a debt of gratitude to Professor Walter Hollister, who really bent over backwards for me to help me get this paper submitted in time to graduate.

Thanks to my father, Gerry, and my mother, Eleanor, for always being there for me, and for not trying to understand too much of what I was doing. I love you both.

Finally, I have to thank Bevin for helping to relieve the stress that always crops up with a project of this scale, and for making me smile. I love you, babe.

* This work was sponsored by the Department of the Air Force under Contract No. F19628-90-C-0002.
This page left intentionally blank
To Lillie Newman

who, at 84, died far too young
# Table of Contents

Chapter 1: Introduction ........................................................................................................ 13  
1.1 Space Surveillance Overview ...................................................................................... 14  
1.2 The Space-Based Visible Sensor .................................................................................. 17  
1.3 Tasks of a Space-Based Sensor .................................................................................... 20  
   1.3.1 New Launch Acquisitions .................................................................................... 21  
   1.3.2 Lost Object Searches ......................................................................................... 21  
   1.3.3 Uncorrelated Target Re-Acquisitions ................................................................. 22  
   1.3.4 Catalog Maintenance ......................................................................................... 22  
1.4 Project Objectives ......................................................................................................... 22  
1.5 Summary of Remaining Chapters ................................................................................ 23  

Chapter 2: Theory ................................................................................................................. 25  
2.1 Scheduling Techniques ................................................................................................. 25  
   2.1.1 Local Scheduling ............................................................................................... 25  
   2.1.2 Global Scheduling ............................................................................................. 26  
   2.1.3 N-Step Look Ahead Scheduling ......................................................................... 27  
2.2 Optimal Search Theory ................................................................................................. 27  

Chapter 3: The Algorithm ..................................................................................................... 31  
3.1 The Orbital Elements ..................................................................................................... 31  
   3.1.1 The Euler Angles ............................................................................................... 31  
   3.1.2 The Orbital Path ............................................................................................... 35  
3.2 The Uncertainty Space ................................................................................................. 37  
   3.2.1 The Normal Probability Distribution .................................................................. 38  
   3.2.2 Generating the Uncertainty Space ....................................................................... 41  
3.3 Searching the Space ...................................................................................................... 44  
   3.3.1 The Value Function ............................................................................................ 44  
   3.3.2 The N-Step Look Ahead .................................................................................... 46  

Chapter 4: The Implementation ............................................................................................. 49  
4.1 Program Features .......................................................................................................... 49  
   4.1.1 Sensor Maneuver Models .................................................................................... 49  
   4.1.2 Constraint Modeling ......................................................................................... 51
Chapter 5: Field Testing

5.1 The Experimental Test System

5.2 Tests Planned

5.2.1 Simple RSO Search

5.2.2 RSO Search With One Element Altered

5.2.3 RSO Search With Two Elements Altered

5.2.4 Debris Re-Acquisition

5.3 Test Execution

5.3.1 May 7th Testing

5.3.2 May 8th Testing

5.3.3 May 9th Testing

5.3.4 May 10th Testing

5.4 Test Results

5.4.1 Acquisition Failures

5.4.2 N Variation Tests

5.4.3 Debris Re-Acquisition Tests

Chapter 6: Analysis

6.1 N-STASOS Code Errors

6.2 Benefits of the N-Step Look Ahead

6.3 Computational Resources

6.4 Analysis Summary

Chapter 7: Conclusions

7.1 Summary

7.2 Recommendations for Further Work

7.2.1 Detectability Model

7.2.2 Use of Multiple CCDs

7.2.3 SBV Script Generation

7.2.4 Improved Constraint Models and Simulator Interface

7.2.5 Multiple Object Searches

7.2.6 Improved Memory Usage
Appendix C: ORBLIB Subroutines

C.1 Subroutine JAMEOE .......................................................... 107
C.2 Subroutine JOEGIC .......................................................... 108
C.3 Subroutine JDSGRC ......................................................... 108
C.4 Subroutine JGRCIC .......................................................... 109
C.5 Subroutine JDDAY ........................................................... 109
C.6 Subroutine JMDAY ........................................................... 110
Chapter 1: Introduction

Starting in 1994, Lincoln Laboratory\(^1\) will have the opportunity to demonstrate operational space-based space surveillance concepts to the defense and space communities using the Space Based Visible (SBV) sensor. The SBV is an optical, angles-only sensor which will be launched into orbit on board the Midcourse Space Experiment (MSX) satellite. The MSX is a Strategic Defense Initiative Organization (SDIO) satellite that will be integrated and operated by the Applied Physics Laboratory at the Johns Hopkins University. SBV surveillance experiments will be controlled by the SBV Processing Operations and Control Center (SPOCC), located at Lincoln Laboratory.

One of the many issues that have arisen in the development of a space-based space surveillance system is how to effectively schedule the use of an orbiting surveillance sensor. Until now, the only scheduling algorithms available for the SBV were taken directly from ground-based sensors, with relatively little modification. While these algorithms are effective for ground-based sensors, they do not consider the unique aspects of space-based space surveillance, which include:

- **Observing geometry:** A space-based sensor in an orbit like the one planned for the MSX travels at over 7,000 meters per second relative to the Earth below, which yields changes in observing geometry more rapidly than in ground-based systems.

- **Resource conservation:** Space-based sensors must rely on battery power for their operation. These batteries can only operate for a limited time before it is necessary

---
\(^1\) A Federally Funded Research and Development Center (FFRDC) administered by the Massachusetts Institute of Technology.
to recharge them. Ground-based sensors, on the other hand, usually have continuous power available. In addition, the thermal conditions of a space-based sensor must be considered, because in order to operate properly, the satellite and sensor alike must be kept within a limited range of temperatures.

These and others peculiarities of space-based space surveillance necessitate the rethinking of current scheduling methods. This project seeks to accomplish that by developing an improved scheduling algorithm for a space-based sensor, and implementing that algorithm for use with the SBV. Since the SBV has not, as yet, been launched, the implementation will be tested on a ground-based optical sensor whose operation is somewhat similar to that of the SBV.

Before moving into the specifics of this project, it is useful to take a look at space surveillance in general, and how it has progressed to the need for a space-based sensor system.

1.1 Space Surveillance Overview

Space surveillance in the United States is the responsibility of the U.S. Space Command (USSPACECOM), which operates the Space Surveillance Center (SSC) at Cheyenne Mountain Air Force Base in Colorado Springs, Colorado. The SSC has a mandate to maintain a catalog of every object orbiting the Earth that is large enough to be tracked by radar or optical sensor. This catalog currently contains approximately 8,000 objects, ranging from active payloads down to debris pieces as small as ten centimeters in diameter [Howell, 1987]. The number of objects in the SSC catalog is constantly increasing for several reasons, including the following:
In 1957, there was only one spacefaring nation: the Soviet Union. In fact, space surveillance became a reality on October 4th of that year, when Sputnik 1 became the first man-made object to orbit the Earth. Today, over 20 nations have orbited satellites, and new payloads are launched every year.

Launch vehicles deliver more to orbit than just the payload. Expended rocket bodies also become part of the orbital environment. Over time, the structural integrity of these rocket bodies can break down, allowing residual fuel and oxidizer to leak. The explosion resulting from their combination can create hundreds of debris pieces.

Most satellite orbits decay over time due to atmospheric drag. This decay is often countered using thrusters, which can only be supplied with a finite amount of propellant. When a satellite's propellant runs out, it often deorbits and can break up as it re-enters the atmosphere. Such breakups produce myriad pieces of orbital debris. In fact, debris from launches and breakups account for some 75 percent of the objects in the SSC catalog.

By the provisions of USSPACECOM Regulation 55-12, the SSC must identify new objects and add them to the catalog [Wilson, 1993]. This task is undertaken both for reasons of national security, and because the increasing number of debris pieces in orbit threatens the operational safety of current and future payloads. In the case of the Space Shuttle and the planned space station, this debris can also threaten human life. One is reminded of a flight of the Space Shuttle Challenger in 1983, during which a paint chip from an unknown satellite carved out a 1/4-inch pit in an Orbiter window [Howell, 1987]. While no lives were ever in danger, the threat definitely exists, and can only increase with time.
The search for new objects and the maintenance of the SSC catalog are currently carried out by the Space Surveillance Network (SSN), which consists of a global collection of radar and optical sites. Lincoln Laboratory has been involved in the SSN since its inception, and has participated in space surveillance since the first satellite launch in 1957 [Wells, 1981]. The laboratory currently operates several space surveillance sensors, including the Millstone Hill Radar in Westford, Massachusetts, the ARPA Long-Range Tracking and Instrumentation Radar (ALTAIR) in the Marshall Islands, and the Experimental Test System (ETS), an optical sensor site on the White Sands Missile Range near Socorro, New Mexico.

The SSN, while historically successful in the acquisition and tracking of spaceborne targets, experiences several operational limitations. In the case of ground-based optical sites, for example, observations can only take place at night, and then only when the skies are clear. In addition, the network as a whole is subject to political and geographical constraints. It is becoming increasingly difficult and costly to build and maintain sensor sites outside of the United States. Therefore, while the SSN has assets around the world, there are gaps in its coverage which limit its ability to support certain time-critical missions, and which can hinder typical catalog maintenance tasking [Dyjak, 1988].

The limitations of the SSN can be reduced by the use of space-based space surveillance sensors in conjunction with, or as an eventual replacement for, the current ground-based network. The SBV sensor represents an opportunity to demonstrate that space-based sensors can be used in the maintenance and continued development of the SSC catalog.
The next section of this document briefly discusses the characteristics of the SBV sensor, and its platform, the MSX satellite.

1.2 The Space-Based Visible Sensor

The Space-Based Visible (SBV) sensor, as shown in Figure 1-1, is a six-inch aperture telescope constructed with an off-axis, three mirror anastigmat. The off-axis design of the telescope actually places the focal plane in a separate chamber from the primary mirror. Because light entering the telescope is directed through a beam stop before it reaches the SBV focal plane, a large portion of the stray light that enters the telescope never reaches the focal plane. Therefore, the SBV can make observations in the bright background conditions that occur near the Earth limb. The optical layout of the telescope is shown in Figure 1-2.

Figure 1-1: The Space-Based Visible (SBV) Sensor
The focal plane of the SBV consists of a 1 x 4 array of Lincoln Laboratory CCID7 charge-coupled devices (CCDs). Each CCD has a 1.4° x 1.4° field of view, and consists of 176,400 pixels (420 x 420), for a total field of view of 1.4° x 5.6° (420 x 1,680 pixels). The signal processor incorporated in the SBV processes data from one CCD at a time. It identifies stars and moving targets (which appear as streaks), and stores the information for later downloading to a ground station [Harrison, 1991].
The SBV sensor is attached to the frame of the Midcourse Space Experiment (MSX) spacecraft, a Strategic Defense Initiative Organization (SDIO) experimental satellite whose mission is to "gather data on targets of interest and backgrounds in wavelengths ranging from the infrared to the ultraviolet" [Harrison, 1991]. It will operate in an 880 kilometer near-sun-synchronous, retrograde orbit that will allow the MSX (and thus the SBV) to observe the entire geosynchronous belt. This is useful for catalog maintenance purposes because a large portion of active payloads are contained within the belt. The MSX is shown with the SBV hardware locations indicated, in Figure 1-3.

Figure 1-3: The Midcourse Space Experiment (MSX) Satellite

In addition to the SBV sensor, the MSX will carry other sensor packages into orbit. These packages are:
- The SPIRIT III, an infrared telescope developed by the Space Dynamics Laboratory at Utah State University. It is cryogenically cooled with solid hydrogen. The cryogen's depletion must be carefully controlled to insure that the operational lifetime of the telescope meets mission specifications. To limit cryogen boil-off, constraints on the operation of the MSX are introduced which prohibit pointing the satellite in such a way that either the SPIRIT III aperture or radiator is exposed to the sun.

- The UVISI, a suite of ultraviolet and visible band sensors developed by the Applied Physics Lab at the Johns Hopkins University.

None of the MSX sensor packages has gimbals or any other method of independent pointing. The SBV and the UVISI sensors are bolted external to the spacecraft bus, while the SPIRIT III is encased within the bus. All three are co-aligned. This means that an attitude maneuver of the SBV (or any other sensor) translates directly into an attitude maneuver of the MSX satellite, and vice-versa. Therefore, pointing constraints affecting one sensor affect every sensor, and it becomes necessary to consider every documented operating constraint or cost for all the sensors while operating the SBV, and thus in the scheduling algorithm developed here. All constraints affecting the MSX satellite and its component sensor packages are documented in the *MSX Operational Constraints and Requirements Handbook* (MOCARH), compiled and distributed by the Applied Physics Lab [Harvey, 1992].

1.3 Tasks of a Space-Based Sensor

In order to develop a scheduling algorithm for the SBV, it is necessary to determine the types of surveillance that the sensor may be tasked to perform. This section discusses typical tasks to be performed by an operational space-based space
surveillance sensor. For reasons of simplicity, the algorithm developed for this project was designed mainly to search for one space object at a time. Therefore, those tasks involving single targets will be more appropriate for this undertaking.

1.3.1 New Launch Acquisitions

When new satellites are launched by foreign powers, the SSN is tasked by the SSC to locate and catalog them as quickly as possible. For launches into high altitude orbits, there may be as many as three opportunities to do this: when the satellite is in a parking orbit, when it is performing an orbital transfer, and when it reaches its final orbit. Most satellites follow typical launch profiles which are known to the SSC, so that element sets for these orbits can be used to generate a search for the new target.

Because a new launch acquisition involves locating a single object, using a small number of predictive element sets, it is an appropriate type of search to be considered for this project.

1.3.2 Lost Object Searches

Occasionally, a catalogued object is not tracked for a long period of time, causing the validity of the catalog element set to decay to a point where it cannot be used to acquire the object without a search. The re-acquisition of such an object is an ideal search for this project, because it involves a single target, with uncertainty on its orbital elements that can be estimated.

A similar situation occurs when a catalogued object maneuvers, thus invalidating its element set. There is a limited number of useful maneuvers a satellite can make, and a
limited number of ways to carry out any such maneuver. Therefore, given the type of satellite, it is usually possible to bound a search space for target re-acquisition.

1.3.3 Uncorrelated Target Re-Acquisitions

When an SSN sensor discovers an object that does not correlate to an object in the SSC catalog, it is tracked long enough to generate an initial element set for the orbit. This element set is used to re-acquire the uncorrelated target (UCT). Like the lost object search, UCT searches are ideal for this project, because the initial element set for the target is uncertain.

1.3.4 Catalog Maintenance

The SSC catalog contains orbital element sets for each object in the catalog. The validity of these orbital elements decreases with time, so these element sets must be continuously updated. To that end, all catalogued objects must be periodically re-acquired and tracked. While catalog maintenance is an important part of the functionality of the SBV, it was decided (as mentioned at the beginning of this section) to focus this study on searches for single objects.

1.4 Project Objectives

Having discussed the background and the motivation for this project, its aims can now be set forth. They are as follows:

- To examine previous research in the fields of scheduling and optimal search theory, and their application to space surveillance.
* To use the results of that research, in addition to the specific requirements of space-based space surveillance, to develop an algorithm to efficiently schedule searches for a space object, given some \textit{a priori} estimate of uncertainty in the orbital elements of the target.

* To implement that algorithm for use in conjunction with Lincoln Laboratory's SBV sensor, but to make the implementation general enough to allow application to other sensors, both ground and space-based.

* To validate the technique and to test the implementation using a ground-based optical sensor (since no space-based sensors are currently available).

The result of this project is the N-Step Ahead Single Object Scheduler, also known as N-STASOS. Coded in the C programming language, N-STASOS uses an N-step look ahead strategy to schedule searches. This strategy and its implementation will be discussed in the following chapters.

1.5 Summary of Remaining Chapters

* \textit{Chapter 2} discusses the scheduling techniques considered in the development of N-STASOS, including current scheduling practices for ground-based sensors. It motivates the choice of an N-step look ahead algorithm and relates search scheduling to optimal search theory.

* \textit{Chapter 3} focuses on the N-step look ahead algorithm at the heart of N-STASOS. It describes the Keplerian orbital element set, and discusses how the elements
are related to one another. It then goes on to show how uncertainties in each orbital element can be used to generate an uncertainty volume in space, and then how best to search that volume to find the desired target.

• Chapter 4 discusses the implementation of N-STASOS. It describes the code, and provides a guide to its use, including a runtime example. The chapter also discusses the testing performed at Lincoln Laboratory in preparation for field trials at the Experimental Test System (ETS) in Socorro, New Mexico.

• Chapter 5 concentrates on testing performed at the ETS. It describes the site, and then discusses the N-STASOS test plan. The chapter concludes with a discussion of the testing results.

• Chapter 6 presents an analysis of the benefits accrued through the use of an N-step look ahead strategy relative to a local strategy.

• Finally, Chapter 7 concludes the document with a summary and recommendations for further work.
Chapter 2:
The Theory

This chapter presents three different scheduling methods considered in the development of N-STASOS: local scheduling (the current standard in ground-based sensor scheduling), global scheduling, and N-step look ahead scheduling. It then motivates the choice of N-step look ahead scheduling and discusses the applicability of optimal search theory to this undertaking.

2.1 Scheduling Techniques

2.1.1 Local Scheduling

Local scheduling is the current standard for scheduling ground-based space surveillance sensors. It is a technique which takes into account only the current sensor and target situations when making scheduling choices. The Millstone Hill Radar currently uses a local scheduler, called the Millstone Dynamic Scheduler (MIDYS), developed by Gerry P. Banner and William F. Burnham [1981] of Lincoln Laboratory. MIDYS has been in operation for over 10 years.

MIDYS considers the current scheduling environment, i.e. the positions of all satellites to be tracked, their visibility and their detectability, to make the next incremental scheduling choice. Each of these satellites is assigned a "figure of merit", based on such factors as historical element set information, target cross-section, and SSC priority. This figure of merit is largest for objects which are most desirable to track next. MIDYS then produces a list of the $K$ best objects to be viewed, where $K$ is a user-selectable number from one to 10. Like many local schedulers designed for ground-based
operation, MIDYS depends on an operator to make the final selection of which object to view next.

Because MIDYS is computationally very efficient, it can run in near-real-time, and thus easily adapt to last-minute changes in observing schedules, sensor outages, or failure to acquire a target. However, since there is no real-time communication with the MSX satellite or the SBV sensor, the entire SBV schedule must be prepared in advance of the uplink opportunity, thus partially obviating these advantages.

The MIDYS algorithm was extended for use with the SBV sensor. The new scheduler, called the Space-Based Dynamic Scheduler (SPADYS), is the current baseline scheduler for the SBV. SPADYS assumes no real-time communication with the sensor. In addition, it circumvents the MIDYS operator choice, automatically selecting the object with the highest figure of merit to track next.

2.1.2 Global Scheduling

While local schedulers consider only the current situation when making incremental scheduling choices, global scheduling strategies consider the scheduling environment for the entire time allotted to a schedule. This environment includes potential sensor locations, sensor capabilities and resource costs, target positions, and constrained regions.

A local scheduler may make good incremental choices, but the complete schedule formed when these choices are assembled may be less than optimal. The additional information available to global schedulers allows them to minimize the missed opportunities that may occur. A properly executed global scheduling process should
yield the most optimal schedule for given search objectives. However, of all the scheduling techniques considered, global scheduling is the most computationally intense. Preliminary tests on a global scheduler developed at Lincoln Laboratory indicate that the amount of time and computer resources necessary to generate a schedule using a truly global approach are prohibitive.

2.1.3 N-Step Look Ahead Scheduling

If local scheduling can potentially miss important viewing opportunities, but global scheduling is too computationally intensive to realistically be used, it is logical to try to find a middle ground between the two strategies.

N-step look ahead scheduling is just such a middle ground, and so was the chosen scheduling technique for this project. It attempts to capture some of the look ahead advantage of global scheduling, while reducing the number of calculations necessary to complete a schedule by limiting to $N$ looks ahead, where $N$ is a small number determined by the user.

Most tests of the N-step algorithm developed for this project limited $N$ to three or less. Preliminary testing indicated that values much higher than three do not provide a significant scheduling advantage (i.e. the likelihood of finding the target does not appreciably increase) when the additional computation necessary is considered.

2.2 Optimal Search Theory

The research for this thesis involves the theory of optimal search, a branch of operations research which has many applications not only in aerospace, but in search and
rescue, drilling for natural resources, law enforcement, and allocation of marketing resources, just to name a few examples.

A comprehensive description of the theory of optimal search was provided by Stone [1975]. He concentrated on the problem of finding a stationary target and briefly discussed the theory's application to moving targets. Stone's text was the basis for most of the other literature considered in the research for this thesis, including a detailed summary of optimal search techniques for moving objects by Brown [1980], Stone's colleague. These techniques were applied mainly to naval searches (for personnel overboard, lost ships, etc.).

Optimal search theory was applied to the problem of satellite search from ground-based sensors by Taff [1976, 1977, 1979, 1982, 1983, 1984, 1989] and Richardson et al. [1984] who made extensive use of Brown's and Stone's research to develop search plans and test them at Lincoln Laboratory's Experimental Test System (ETS), in Socorro, New Mexico. In addition, Stone [1983] presented several approaches to search planning that are currently used elsewhere.

Most optimal searches use a Bayesian approach. The position or state vector of the target (moving or stationary) is assumed to have an *a priori* probability distribution. Naturally, this distribution is different for different types of targets and different situations. In the case of moving targets, the probability distribution changes with time.

In addition, the sensor is assumed to have a detection function which determines the likelihood that the sensor actually detects the target when it points at its location. This is often modelled as an exponential function of time, such as:
\[ b(t) = 1 - e^{-t} \]  

where \( t \) is time, expressed in hours. Like this example, the function is usually defined such that the likelihood of detection increases with observation time. Note that Equation 2-1 is merely an example. Detection functions can vary greatly from sensor to sensor. In the case of the SBV sensor, the function is determined by the parameters of the on-board signal processor, the optics of the sensor, and the camera.

The detectability of a sought object differs from the detection function in that it depends only on the condition of the target, and not of the sensor. It is related to the illumination of the sought object relative to the background radiation and to its cross-section. The detectability is often expressed as a percentage, where the higher the value, the brighter and/or larger the target.

It is also assumed that the allocation of search effort has some cost, such as time or money (or use of satellite resources, for the SBV/MSX). The optimal search is the one that, given the probability distribution of the target's position and the characteristics of the sensor, maximizes the probability of finding the target, subject to the constraint that the search's total cost must not exceed a given value.

Of course, "optimal" is only in the eye of the beholder. One might specify that the optimal search minimizes time to acquisition regardless of cost, or that it minimizes cost regardless of detection probability. These priorities are quantified through the use of an objective function.
The objective function is a tool for representing scheduling goals. It can be used to determine the effectiveness of a given schedule. For the case of a search, the objective function might look like the following:

\[ f_{\text{obj}} = w_p p + w_r r - w_n n \]  

(2-2)

where \( p \) is the probability of acquiring the target, \( r \) is the time available to operate with remaining renewable resources (i.e. battery power) after the completion of the search, and \( n \) is the cost of using non-renewable resources (i.e. cryogen). The values \( w_p, w_r, \text{and } w_n \) are weights for each of the parameters. The best schedule is the one which maximizes the objective function. In this example, note that increasing probability of acquisition increases the value of the objective function, and so represents a better schedule. Increasing cost, on the other hand, decreases the function, representing a less optimal schedule.

The application of search theory, to the particular problem considered here, is made more complex by the constraints of a space-based sensor. The optimal search plan may state that at time \( t \), the sensor should be pointed towards the sun because that is the most likely place for the target to appear. However, the SBV cannot be pointed at the sun due to attitude constraints on the MSX satellite. In addition, the SBV may be pointed at the right place to observe a particular object, but it will not be detected because it is not illuminated. The constraints unique to a space-based sensor make it unlikely that the algorithm developed here will be truly optimal. It is merely intended to offer the best chance of detection, providing practical, if not theoretical, optimality.
Chapter 3:
The Algorithm

This chapter discusses the details of the N-step look ahead algorithm at the heart of the N-Step Ahead Single Object Scheduler. The first part of this chapter discusses the Keplerian orbital element set and the relationships among the orbital elements. The second section then discusses how the orbital element set is used to generate an uncertainty space, and the third section presents how that space is searched to produce a schedule.

3.1 The Orbital Elements

Satellites in orbit around the Earth travel in elliptical paths. These orbits are identified by their orbital elements. Element sets are catalogued and stored by tracking systems so that they can be used to find a particular orbiting object at a later time. The orbital elements most often used by Lincoln Laboratory tracking systems are: inclination \(i\), right ascension of the ascending node \(\Omega\), argument of perigee \(\omega\), eccentricity \(e\), semi-major axis \(a\), mean anomaly \(M\), and an epoch at which these elements are recorded \(t\). This section defines these orbital elements, and discusses the relationships among them.

3.1.1 The Euler Angles

To show how the orbital elements determine a satellite's orbit, it is first necessary to establish a frame of reference from which the orbit can be described, using the three Euler angles: the right ascension of the ascending node \(\Omega\), the inclination \(i\), and the argument of perigee \(\omega\).
Figure 3-1 shows an equatorial Earth-Centered Inertial (ECI) coordinate frame. The $x$-axis of this coordinate frame points towards the vernal equinox at the current epoch, also known as the first point of Aries ($Y$). This point is along the line which describes the intersection of the Earth's equatorial plane with the plane of the ecliptic. The $y$-axis is 90 degrees to the east of the $x$-axis, in the Earth's equatorial plane. The $z$-axis points towards the Earth's physical north pole, thus forming a right-handed set of axes. This coordinate frame can now be rotated, using the Euler angles, to define the orbital plane.

![Figure 3-1: Equatorial ECI Coordinate Frame](image)

Staying in the equatorial plane, the axes are first rotated east through the angle $\Omega$, thus establishing the direction of the line of nodes (the $n$-axis in Figure 3-2). The
ascending node is the point at which a satellite crosses the equatorial plane traveling in the +z direction. This satellite later crosses the plane again in the opposite direction at the descending node.

The inclination is defined as the angle between the equatorial plane and the plane of the satellite’s orbit. Rotating the coordinate frame by the right-handed rule about the n-axis through an angle of \( i \) establishes the orbital plane of the satellite. The rotated z-axis is now called the h-axis, because it now points perpendicular to the orbital plane, in the direction of the angular momentum \( (h) \) of the satellite. The twice-rotated y-axis is now called the m-axis, as shown in Figure 3-2.

![The Orbital Frame](image)

**Figure 3-2: The Orbital Frame**

The next step is to rotate the coordinate frame east in the orbital plane through an angle of \( \omega \). This establishes the e-axis, which points to the perigee of the orbit, that point
at which the satellite makes its closest approach to the Earth. The rotated m-axis now points in the direction of the parameter of the orbit (also known as the semi-latus rectum of the ellipse), and so is called the p-axis. Note that this coordinate frame is still right-handed.

These three Euler angles establish a useful inertial coordinate frame in the plane of the satellite's orbit, as shown in Figure 3-3. The remaining elements establish the shape of the orbital path and the satellite's position within that path.

Figure 3-3: The Euler Angles
3.1.2 The Orbital Path

The shape of the orbital ellipse is established via two of the orbital elements, $a$ and $e$. As shown in Figure 3-4, $a$ represents the semi-major axis of the orbit. The semi-minor axis, $b$, is determined from the following equation:

$$b = a\sqrt{1 - e^2}$$  \hspace{1cm} (3-1)

Given this information, the orbital ellipse can be drawn with the Earth at the occupied focus ($F$) and the perigee and apogee distances given by:

$$r_\pi = a(1 - e)$$  \hspace{1cm} (3-2)

$$r_\alpha = a(1 + e)$$  \hspace{1cm} (3-3)

The position of the satellite in the orbital path, also shown in Figure 3-4 is determined by the use of the orbital element $M$, the mean anomaly. The mean anomaly represents what the angular position of the satellite would be if it were traveling along a circle of radius $a$ at a constant angular rate.

As the figure shows, the mean anomaly does not yield the actual angular position of the satellite. It is used because it is easily calculable and its time derivative is a constant. The mean anomaly is given by:

$$M = n(t - t_o)$$  \hspace{1cm} (3-4)

where $t_o$ is the time of perigee passage and $n$, called the mean motion, is given by:
\[ n = \sqrt{\frac{\mu}{a^3}} \]  

(3-5)

with \( \mu \) being the gravitational constant (\( = 398,601.2 \text{ km}^3/\text{sec}^2 \)).

Another angle commonly used is the eccentric anomaly, shown as \( E \) in Figure 3-4.

It is related to the mean anomaly by Kepler's equation:
\[ M = E - e \sin E \] \hspace{1cm} (3-6)

The eccentric anomaly is measured from the center of a circle of radius \( a \) circumscribed about the orbital ellipse. Unlike the mean anomaly, it does not change at a constant rate. The position of the satellite in orbit at a particular time, given its eccentric anomaly, can be determined graphically by dropping a perpendicular line to the semi-major axis of the orbit. The point at which this line intersects the orbital ellipse is the position of the satellite.

The true anomaly, \( f \), represents the angular position of the satellite measured from the occupied focus of the ellipse (the center of the Earth, in this case). Its time rate of change is such that the orbital radius sweeps out equal area with equal time (Kepler's second law). It is related to the eccentric anomaly by the following equation:

\[ \tan \frac{f}{2} = \frac{1 + e}{1 - e} \tan \frac{E}{2} \] \hspace{1cm} (3-7)

The uncertainty space generated by N-STASOS is maintained in orbital element space. The next section describes how the orbital elements are used to generate that uncertainty space.

3.2 The Uncertainty Space

N-STASOS takes as input the mean element set for a target, and \textit{a priori} uncertainties on each of the six orbital elements, \( i, \Omega, \omega, e, a, \) and \( M \), as determined by the user. These uncertainties are used to generate a six-dimensional volume which is to be searched. It is advantageous to maintain this volume in six-dimensional form, because the uncertainty space will not evolve with time. This would not be the case if the volume
were maintained in Right Ascension / Declination space. Since the target is in motion, both the position and shape of the distribution will evolve with time in three-dimensional space.

This section discusses how the N-STASOS algorithm generates the uncertainty space necessary to find the desired object. The probability distribution used is first described, and then its use in the generation of the uncertainty volume is discussed.

3.2.1 The Normal Probability Distribution

Every uncertain orbital element for an N-STASOS search is assumed to be normally distributed, with a standard deviation on each element given by the user. If an element is normally distributed, the probability for any given value of the element is given by the following equation:

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

(3-8)

where \( \mu \) is the mean of the variable in question and \( \sigma \) is the standard deviation. This distribution, also known as a Gaussian distribution, is an even function, as shown in Figure 3-5. In the figure, \( \mu=0 \).
The normal distribution is a two-tailed, symmetric function representing the probability that \( x \) will be a particular value. The total area of the curve is equal to one, i.e.:

\[
\int_{-\infty}^{\infty} f(x) \, dx = 1
\]  

(3-9)

and more than 99 percent of the area of the curve is contained within three standard deviations of the mean value.

Depending on the orbit of a given satellite, a one-tailed distribution such as a Rayleigh may be more appropriate. For example, in lower orbits, drag is more likely to be a factor. Drag will cause a satellite to drop in altitude, thus increasing its angular rate.
Therefore, the object's mean anomaly would more likely be greater than the nominal value.

The Rayleigh distribution, as shown in Figure 3-6, is characterized by the following equation:

\[
f(x; \alpha) = 2\alpha x e^{-\alpha x^2}
\]  

(3-10)

where \(\alpha\) is a constant, and \(\alpha\) and \(x\) are both greater than zero.

In this implementation of N-STASOS, a normal probability distribution was used for all orbital elements, because it was easy to work with. The modular, subroutine-oriented design of N-STASOS would make it a simple matter to change the type of distribution, should future research determine that the Rayleigh or some other distribution is more appropriate.
3.2.2 Generating the Uncertainty Space

N-STASOS takes as input a nominal element set for the object being sought and the uncertainty for each orbital element. This uncertainty is given as a standard deviation on a normal probability distribution, and is determined by the type of search. For example, an along-orbit search would have uncertainty in the mean anomaly of the desired target, while a maneuver search might have uncertainty both in the inclination and the right ascension of the ascending node of the target.

A probability distribution with values calculated at discrete intervals is generated in orbital element space using the standard deviations. The granularity (or interval between samples) of this distribution has been chosen such that after the transformation
to three-dimensional space, the grid size is appreciably smaller (in Right Ascension and Declination) than the sensor field of view. In the current implementation, angular elements are calculated at an interval of 0.1°. This interval can be modified, as appropriate to any given circumstance.

To generate a discrete uncertainty space over time, N-STASOS assumes an average complete look time, $t_{avg}$ (which includes maneuver time, settle time, data collection, and signal processing), and assumes that each look begins $t_{avg}$ seconds after the previous look. For the SBV, this is a good assumption, due to the relatively high settle time of the spacecraft. It is also a valid assumption for ETS, because for most searches, the maneuver time is very small relative to the time required to collect data. The algorithm converts the distribution in orbital element space to sensor-centered Right Ascension / Declination space for $N$ snapshots, each $t_{avg}$ seconds apart.

This RA/DEC map contains the possible positions of the desired object modelled from the probability distributions on each of the uncertain elements. The positions are calculated by propagating element sets consisting of all possible combinations of uncertain elements to the specified times. Each element set (also known as a "pseudo-object") has an associated probability of appearance (i.e. of being the actual element set for the desired object) which is the product of the probabilities of the object having a given value for each uncertain element. For example, the probability of appearance of element set $k$ is:

$$P(k) = P(i_k \pm \Delta i_k) \cdot P(\Omega_k \pm \Delta \Omega_k) \cdot P(\omega_k \pm \Delta \omega_k)$$

$$\cdot P(e_k \pm \Delta e_k) \cdot P(a_k \pm \Delta a_k) \cdot P(M_k \pm \Delta M_k)$$

(3-11)

where $i$, $\Omega$, $\omega$, $e$, $a$, and $M$ are the Keplerian orbital elements described earlier, the $\Delta$'s represent the granularity of each element used to generate a discrete probability.
distribution. Also, the probability of an element which is not uncertain is exactly equal to one.

Once the snapshot maps are generated, the individual pseudo-object positions are collected into overlapping squares which approximate the field of view of the sensor. In the current implementation, each of these squares is $1^\circ \times 1^\circ$. This size (compared to the $1.4^\circ \times 1.4^\circ$ field of view of the SBV) was chosen so that edge effects can be ignored, and also so that the field of view considered does not depend on the roll angle of the satellite. As shown in Figure 3-7, for any roll angle, $\theta$, the square still fits inside a $1.4^\circ \times 1.4^\circ$ square.

![Diagram showing sensor field of view independent of roll angle](image)

**Figure 3-7: Sensor Field of View Independent of Roll Angle**

In this implementation, each of the squares is separated by $0.1^\circ$ in space. This separation can be adjusted by the user.
3.3 Searching the Space

This section focuses on how the N-STASOS algorithm uses the information about the uncertainty space to determine where to point the sensor, and thus, generate a search schedule. A value function, or figure of merit, is introduced, and its use in schedule generation and evaluation is explained.

3.3.1 The Value Function

Given the probability distribution in RA/DEC space, it is logical to assume that the best schedule would point to the area with the highest concentration of probability first. However, it is not always quite that simple. The probability of appearance of the target, the initial condition of the sensor, and other factors must be combined into a figure of merit, or value function. The other factors to be considered include the following:

- **Likelihood of detection:** It is not enough for an object to be in a particular location, but it also must be bright enough to be seen, and must be in that location long enough to be recognized by the sensor. The likelihood of detection is the probability that the object will be recognized if it is in the sensor field of view. It is often expressed as a percentage, where zero percent means that the target will not be recognized, even if the sensor is staring right at it, and 100 percent indicates that the object will definitely be recognized.

- **Time to maneuver:** The time to maneuver the sensor between two points is an important part of the value function. A distant point may have a relatively high probability concentration, but larger maneuvers take a correspondingly long time to
complete, and can consume a large amount of power. This time and these resources may be better spent searching other areas.

- **Violation of pointing constraints:** If a desirable location is affected by a pointing constraint, the cost of violating that constraint must be accounted for. In the case of a hard constraint, which cannot be violated under any circumstances, the value of the constrained location would always be set to zero.

- **Use of sensor and satellite resources:** Resource allocation is a critical issue for a space-based sensor, since its energy is supplied by batteries which last a limited amount of time before a recharge is required. In addition to power, other resources which are not renewable must be considered. In the case of the MSX satellite, solid hydrogen cryogen is used to cool one of the sensors on-board. Some maneuvers may cause the cryogen boil-off rate to increase, thus reducing the lifetime of that sensor. In addition, the heat capacity of everything on board the satellite is a resource, since the temperature of any given piece of the MSX satellite can affect the operation of all of the sensors on board. These resources and others must all be considered in the value function.

Now that all of the necessary inputs to the figure of merit have been introduced, the value function itself can be constructed. In general, it is a weighted sum of the above inputs, with the exception of the probability of appearance and likelihood of detection, which are multiplied before they are summed with the other inputs. The value function is of the following form:

\[ f_{value} = C_{hard} \cdot \{w_{prob} \cdot (P_{appear} \cdot P_{detect}) + w_{time} \cdot t_{maneuver} + w_{resources} \cdot R_{used} + \ldots \} \]  

(3-12)
In Equation 3-12, $P_{\text{appear}}$ is the probability of appearance, $P_{\text{detect}}$ is the likelihood of detection, $t_{\text{maneuver}}$ is the time to maneuver between two points, $R_{\text{used}}$ is the amount of sensor and satellite resources used, and $C_{\text{hard}}$ is a multiplier which is zero if a point is hard constrained (i.e. there is a constraint which cannot be violated at any cost), and is one otherwise. The $w$'s in the equation are weights for each of the variables. These weights are chosen such that increasing probability of appearance and/or detection increases the value function, while increasing maneuver time and resource usage decrease the value.

### 3.3.2 The N-Step Look Ahead

To determine where to look over the current step and the next N steps ahead, the value function is integrated over the N+1 looks, and the schedule with the largest integrated value is chosen as a first cut. Figure 3-8 shows the generation of the snapshot RA/DEC maps which are used to bucket RA/DEC space into 1° x 1° squares. The squares are assigned figures of merit from the above value function, and schedules are generated, each with a corresponding integrated figure of merit (marked SUM in the figure).
Each step of the first-cut $N$-step ahead schedule is then examined to determine how much time should be spent at each calculated position. For example, it may be advantageous to stay at a given position longer than originally planned to allow more of the probability distribution to drift through the sensor field of view. The algorithm considers this possibility for each of the $N$ steps and updates the looks accordingly.

Once the $N$-step ahead schedule is fine tuned, the initial step is considered frozen, and the area of consideration moves to steps 2 through $N+1$, as shown in Figure 3-9. The $N+1$st snapshot (which occurs $t_{avg}$ seconds after the $N$th snapshot) is calculated and
appended to the first-cut N-step schedule, which in turn is recalculated to account for the new information. The process continues until a complete fine tuned schedule is generated for the time allowed.

Figure 3-9: Second Pass Schedule Generation
Chapter 4: The Implementation

The N-Step Ahead Single Object Scheduler has been implemented on a Silicon Graphics Indigo workstation running under the IRIX operating system. It was coded in the C programming language, conforming to the current standard set by the American National Standards Institute (ANSI). ANSI compliance will allow N-STASOS to be ported to other systems with little or no modification. This chapter describes the features of the implementation, discusses local testing, and presents a runtime example of the use of N-STASOS.

4.1 Program Features

This section discusses specific features of the N-STASOS code, including the models used to approximate sensor maneuvers, and methods of constraint modeling. A brief description of the function of each N-STASOS subroutine is included in Appendix A, and diagrams of N-STASOS program flow can be found in Appendix C.

4.1.1 Sensor Maneuver Models

Two maneuver models have been included in N-STASOS: one appropriate for a ground-based sensor, and one for a space-based sensor. The ground-based model is taken from Taff [1982]. It states that the minimum time for a sensor to maneuver between two points separated by an angle, $\Phi$, is:

$$T_{\text{min}} = \sqrt{\frac{2(\alpha + \delta)}{\alpha \delta}} |\Phi|$$

(4-1)
where $\alpha$ and $\delta$ are acceleration and deceleration of the sensor mount, respectively (degrees per second per second), and $\Phi$ is expressed in degrees.

In the case of the telescopes at ETS, the acceleration and deceleration of the mounts are the same (2 degrees per second per second, with a maximum speed of 4 degrees per second). Therefore, Equation 4-1 can be more simply expressed as:

$$T_{\text{min}} = \sqrt{\frac{4}{\alpha} |\Phi|}$$

(4-2)

The following maneuver model for a space-based sensor is currently used for planning experiments with the MSX satellite [Landshof, 1992]. As shown in Table 4-1, maneuvers of less than 2.0 degrees assume a fixed maneuver duration. Larger maneuvers are expressed as a function of the square root of the angular maneuver length.

<table>
<thead>
<tr>
<th>Maneuver Length, Degrees ($\Phi$)</th>
<th>Maneuver Duration, Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 - &lt; 0.4</td>
<td>15</td>
</tr>
<tr>
<td>0.4 - &lt; 1.0</td>
<td>20</td>
</tr>
<tr>
<td>1.0 - &lt; 2.0</td>
<td>25</td>
</tr>
<tr>
<td>2.0 - 60.0</td>
<td>$7 + 10.75\sqrt{\Phi}$</td>
</tr>
<tr>
<td>60.0 - 180.0</td>
<td>$11.625\sqrt{\Phi}$</td>
</tr>
</tbody>
</table>

Table 4-1: Space-Based Sensor Maneuver Model

In addition to the maneuver duration shown in Table 4-1, N-STASOS includes a fixed sensor settle time of 30 seconds, regardless of maneuver length.
4.1.2 Constraint Modeling

Most of the pointing constraints imposed on the MSX satellite are expressed as a "keep-out" zone of a given angular width with respect to the sun, the moon, or the Earth. For example, the sun should be kept outside of a 20 degree half-angle cone centered about the +x-axis of the spacecraft (the boresite of the SBV). Ground-based sites like the ETS also have keep-out zones for the sun and the moon, as well as limits on the elevation angle of the sensor. N-STASOS will not allow a point to be observed if it is in one of these keep-out zones.

To determine whether or not a constraint would be violated at a given RA/DEC point, N-STASOS calculates a unit vector in the direction of that RA/DEC point, and a unit vector pointing to the constraining object (the sun, in this example). N-STASOS then takes the dot product of the two vectors, and thus calculates the angle between them. If that angle is less than the half-angle of the keep-out zone, the point is disallowed.

N-STASOS can maintain keep-out zones for the sun, the moon, and the Earth. The current keep-out zones for the SBV and for ETS are kept in an include file which is accessed by N-STASOS. This file can be modified by the user.

The software is currently limited to the consideration of keep-out zones which affect the boresite axis of the sensor. For the SBV, constraints affecting other axes require knowledge of the sensor roll angle, which is not included in the current implementation.
4.2 Local Testing

Two sets of tests were performed at Lincoln Laboratory to ensure that the N-STASOS program functions properly. The first test was to generate a script for an object with zero uncertainty, and compare the position of that object with its position as calculated by STARS, a validated, operational orbital mechanics software package developed at Lincoln Laboratory [Rozier, 1991]. The zero uncertainty test was performed for several geosynchronous and low altitude objects. In all cases, the right ascension and declination values calculated by N-STASOS differed from STARS by no more than 5/100 of 1°.

The second test was to generate an N-STASOS script for an object with uncertainty in one orbital element, and determine whether or not the object would be found using the generated schedule. This determination was made by calculating the position of the actual object (with a zero uncertainty element set) over the scheduled time using STARS, and then seeing if the generated schedule placed the object within the sensor field of view at some point during the schedule. N-STASOS performed successfully for all objects tested.

4.3 Program Example

The following is a runtime example of N-STASOS, depicting an eight-minute long search for a geosynchronous object with a five degree uncertainty in mean anomaly, using an ETS sensor. In this particular case, the object sought was found on the second of ten looks. Note that user input is shown in boldface type.
The user first starts the program with an input file name. Files beginning "els" are element sets. The user can also enter a file name beginning with "cat", which will cause N-STASOS to read in a file containing a pre-generated element set catalog. If the user does not enter in a file name, the program asks the user to input each element for the mean element set.

```
nstasos els_20945.dat
Welcome to N-STASOS. Time:
Fri Apr 30 16:33:46 EDT 1993
Element set retrieved from file "els_20945.dat".
```

Here, the user enters the uncertainties. Only mean anomaly is uncertain.

```
Enter standard deviations for requested elements.
If value is certain, enter 0.0
Inclination (degrees):  0.0
Right Ascension of the Ascending Node (degrees):  0.0
Eccentricity:  0.0
Argument of Perigee (degrees):  0.0
Mean Anomaly (degrees):  5.0
Semi-Major Axis (earth radii):  0.0
```

The user chooses the number of standard deviations for the search. On a normal probability distribution, a value of three or greater insures that more than 99 percent of the distribution will be covered.

```
Enter number of standard deviations for search:  3
```

Note that the user can save the catalog to a file, so that it does not have to be recreated on a later program execution.

```
Save catalog to file? [YyNn]  Y
Enter file name:  cat_20945.dat
Catalog written to file "cat_20945.dat"
```
The user next chooses the type of site. In the current implementation, the ground-based sites available are the ETS-A dome and the ETS-B dome.

Ground-Based or Space-Based Site? [GgSs]  G
A-Dome or B-Dome? [AaBb]  A
Read site location from file "gdb_etsa.dat".

The user enters the start time, which is converted to days to epoch J2000.

Enter schedule start time:
Calendar format or Orblib format? [CcOo]  C
Months or Days? [CcOo]  D
Year [0-99]:  91
Day [0-365]:  310
Hour [0-24]:  9
Minute [0-60]  36
Second [0-60]  0
Converted to -2979 + 0.900000 (J2000).

The search duration, initial position, average time, and number of steps to look ahead are entered by the user.

Enter search duration (minutes):  8
Enter sensor position at start of search:
Right Ascension (degrees):  0
Declination (degrees)  0
Enter average time for one look (seconds):  45
Enter the number of steps to look ahead (N):  3

Input parameters as follows:
Start time (J2000) = -2979 + 0.900000
Duration (minutes) = 8.000000
Average look time (seconds) = 45.000000
N = 3, 1 uncertain element, 301 pseudo-objects
Initial sensor RA/DEC = 0.000000, 0.000000
Schedule will contain 10 looks.
N-STASOS generates snapshots for the current look plus \( N = 3 \) looks ahead. It then buckets the pseudo-objects and finds the bucket which maximizes the value over the \( N + 1 \) steps. The process is repeated for all 10 looks.

Map frame #1 at epoch -2979 + 0.900000 complete.
Map frame #2 at epoch -2979 + 0.900521 complete.
Map frame #3 at epoch -2979 + 0.901042 complete.
Map frame #4 at epoch -2979 + 0.901562 complete.
Range for frame #1: RA = (31.32, 64.75), DC = (-5.44, -5.23)
Range for frame #2: RA = (31.51, 64.94), DC = (-5.44, -5.23)
Range for frame #3: RA = (31.70, 65.13), DC = (-5.44, -5.23)
Range for frame #4: RA = (31.89, 65.31), DC = (-5.44, -5.23)

Assigning look #1.
Look #1: Best choice (47.84103, -5.19040), value = 4.6003.

Map frame #1 at epoch -2979 + 0.904687 complete.
Map frame #2 at epoch -2979 + 0.905208 complete.
Map frame #3 at epoch -2979 + 0.905729 complete.
Map frame #4 at epoch -2979 + 0.906250 complete.
Range for frame #1: RA = (33.02, 66.44), DC = (-5.44, -5.23)
Range for frame #2: RA = (33.20, 66.63), DC = (-5.44, -5.23)
Range for frame #3: RA = (33.39, 66.82), DC = (-5.44, -5.23)
Range for frame #4: RA = (33.58, 67.00), DC = (-5.44, -5.23)

Assigning look #10.
Look #10: Best choice (54.25206, -5.18985), value = 2.8806.

The program prints the coarse schedule, and gives the user the option of fine tuning it before it is printed to a script file.

Coarse schedule as follows:
Look # 1: RA = 47.841, DC = -5.190, FOM = 4.60, time = 43.81
Look # 2: RA = 47.125, DC = -5.190, FOM = 4.53, time = 43.01
Look # 3: RA = 49.121, DC = -5.190, FOM = 4.52, time = 43.53
Look # 4: RA = 50.212, DC = -5.190, FOM = 4.30, time = 42.31
Look # 5: RA = 46.584, DC = -5.190, FOM = 4.26, time = 41.84
Look # 6: RA = 51.592, DC = -5.190, FOM = 3.90, time = 41.65
Look # 7: RA = 45.956, DC = -5.190, FOM = 3.86, time = 41.26
Look # 8: RA = 52.972, DC = -5.190, FOM = 3.40, time = 41.09
Look # 9: RA = 45.328, DC = -5.190, FOM = 3.35, time = 40.78
Look #10: RA = 54.252, DC = -5.190, FOM = 2.88, time = 45.00

Fine tune schedule? [YyNn] Y
Now fine tuning schedule.
Final schedule as follows:
Look # 1: RA = 47.841, DC = -5.190, FOM = 3.80, time = 32.56
Look # 2: RA = 47.125, DC = -5.190, FOM = 4.53, time = 43.01
Look # 3: RA = 49.121, DC = -5.190, FOM = 4.52, time = 43.53
Look # 4: RA = 50.212, DC = -5.190, FOM = 4.30, time = 42.31
Look # 5: RA = 46.584, DC = -5.190, FOM = 4.26, time = 41.84
Look # 6: RA = 51.592, DC = -5.190, FOM = 3.91, time = 41.65
Look # 7: RA = 45.956, DC = -5.190, FOM = 3.86, time = 41.26
Look # 8: RA = 52.972, DC = -5.190, FOM = 3.39, time = 41.09
Look # 9: RA = 45.328, DC = -5.190, FOM = 3.35, time = 40.78
Look #10: RA = 54.252, DC = -5.190, FOM = 2.88, time = 45.00
Enter file name for schedule script: SCRIPT.0430.5
Schedule written to file "SCRIPT.0430.5".

Schedule complete. Time:
Fri Apr 30 16:44:37 EDT 1993

The script file generated by N-STASOS for this case to command the ETS sensor looks like this:

tcs radec 3 11 21.847490 -5 11 25.442171
tcs wait settle
sleep 32
tcs radec 3 8 30.044405 -5 11 25.223934
tcs wait settle
sleep 55
tcs radec 3 16 28.934636 -5 11 25.005690
tcs wait settle
sleep 45
tcs radec 3 20 50.937957 -5 11 24.787440
tcs wait settle
sleep 43
tcs radec 3 6 20.310264 -5 11 24.569187
tcs wait settle
sleep 44
tcs radec 3 26 22.156155 -5 11 24.350933
tcs wait settle
sleep 44
tcs radec 3 3 49.558033 -5 11 24.132681
tcs wait settle
sleep 44
tcs radec 3 31 53.374139 -5 11 23.914432
tcs wait settle
sleep 44
tcs radec 3 1 18.805772 -5 11 23.696189
tcs wait settle
sleep 44
tcs radec 3 37 0.493390 -5 11 23.477955
tcs wait settle
sleep 49

In the above script file, the first set of three numbers in the `radec` command are the right ascension to be pointed to, expressed in hours, minutes, and seconds. The second set of three numbers are the declination to be pointed to, in degrees minutes in seconds. The `sleep` command tells the sensor how much time (in seconds) to wait before maneuvering to the next position.

In the current implementation of N-STASOS, schedule scripts can only be written in ETS format. It should be a simple matter to add a subroutine that generates SBV pointing commands, if desired.
This page left intentionally blank
Chapter 5:
Field Testing

This chapter discusses the field testing of the N-Step Ahead Single Object Scheduler which took place from May 7th to May 10th, 1993 at the Experimental Test System (ETS) near Socorro, New Mexico. The first section describes the ETS and its history. The second part discusses the test cases generated for this effort and is followed by a section describing how these tests were executed. The final section presents the results of the tests.

5.1 The Experimental Test System

A 1972 Lincoln Laboratory study suggested that deep space surveillance tasks could be carried out more efficiently with optical sensors than with the radars used at that time. To validate this theory, the Experimental Test System was built in 1975 as a test bed for the Ground-Based Electro-Optical Deep-Space Surveillance (GEODSS) program. The ETS served as an operational Space Surveillance Network site until 1982, when the first operational GEODSS site came on-line. Since then, the ETS has been used in many development programs, and contributes to special undertakings such as the Debris Measurement Program.

The ETS is situated in the northwest corner of the White Sands Missile Range, near Socorro, New Mexico. The site was chosen because it offers dry, clear weather and dark skies. (The nearest major population center, Albuquerque, is over 80 miles away.) There are two operational domes at the site, ETS-A and ETS-B, separated by approximately 60 meters. The mount in each dome can support a variety of sensors in both the visible and infrared bands [Gibson, 1991].
For the N-STASOS tests, a visible band sensor was used with a one degree field of view (measured diagonally), and an aspect ratio of 4 : 3 (like a television). The field of view can be zoomed in to one-half of one degree and out to two degrees. All tests used the one degree field of view because it best simulates the size of a single CCD on the SBV focal plane.

5.2 Tests Planned

Four different types of tests were prepared to validate N-STASOS. In increasing order of difficulty, they were: (1) a simple Resident Space Object (RSO) search, (2) an RSO search with one orbital element altered, (3) an RSO search with two orbital elements altered, and (4) the re-acquisition of a new debris piece found by the ETS. Each of these tests is described in the following sub-sections. The space objects used for testing are shown in Table 5-1.
Table 5-1: N-STASOS Test Objects

<table>
<thead>
<tr>
<th>SSC #</th>
<th>Name</th>
<th>Country</th>
<th>Type</th>
<th>Orbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8585</td>
<td>CTS A</td>
<td>Canada</td>
<td>Inactive Payload</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>14189</td>
<td>GPS 07</td>
<td>USA</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>15235</td>
<td>SBS 4</td>
<td>USA</td>
<td>Active Payload</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>15237</td>
<td>Telstar 3C</td>
<td>USA</td>
<td>Active Payload</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>15271</td>
<td>USA 05</td>
<td>USA</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>19216</td>
<td>OSCAR 13</td>
<td>USA</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>19713</td>
<td>Molniya 3-34</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>19751</td>
<td>Cosmos 1989</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>20026</td>
<td>Cosmos 2024</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>20583</td>
<td>Molniya 1-77</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>20646</td>
<td>Molniya 3-38</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>20696</td>
<td>Cosmos 2085</td>
<td>CIS</td>
<td>Rocket Body</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>20742</td>
<td>Molniya 1-78</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>20962</td>
<td>GOES 1</td>
<td>USA</td>
<td>Kick Motor</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>22035</td>
<td>Cosmos 2198</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22037</td>
<td>Cosmos 2200</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22038</td>
<td>Cosmos 2201</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22039</td>
<td>Cosmos 2202</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22054</td>
<td>Soyuz TM-15</td>
<td>CIS</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22161</td>
<td>Freja 1</td>
<td>Sweden</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22195</td>
<td>LAGEOS II</td>
<td>Italy</td>
<td>Active Payload</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>22205</td>
<td>Galaxy VII</td>
<td>USA</td>
<td>Active Payload</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>22231</td>
<td>USA 85</td>
<td>USA</td>
<td>Active Payload</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>22314</td>
<td>TDRS F</td>
<td>USA</td>
<td>Active Payload</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>22581</td>
<td>Unknown Object</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>97121</td>
<td>New ETS UCT</td>
<td>Unknown</td>
<td>Debris</td>
<td>Low Altitude</td>
</tr>
</tbody>
</table>

All tests performed were blind, meaning that the test cases were generated by other Lincoln personnel. The author did not know what biases were added to the test
element sets. The standard deviation of the biases generated using a normal probability distribution was known.

5.2.1 Simple RSO Search

The simple RSO search is intended to check that the N-STASOS coordinate frames agree with those used at ETS, and that N-STASOS can locate an object for which there is no uncertainty in its orbital elements. The test consists of a single look, and is considered a success if the target passes through the sensor field of view within the time allotted.

5.2.2 RSO Search With One Element Altered

The first test of the functionality of the N-STASOS algorithm is to change a single orbital element of an RSO, and then use that altered element set to generate a search schedule. Test cases were generated that placed a bias on one of the following: the semi-major axis, the mean anomaly, the right ascension of the ascending node, or the inclination. This bias was created by a random number generator using a standard deviation of three degrees for the angular elements, and 30 kilometers for the semi-major axis. The test script generated by N-STASOS consists of a set of several sensor looks, and the test is considered a success if the target passes through the sensor field of view at least once during the execution of the test script.

5.2.3 RSO Search With Two Elements Altered

This test uses an RSO element set with two altered elements to create a search schedule. The test cases generated were for a coplanar bias, with random uncertainties in
the argument of perigee (standard deviation of one degree) and the mean anomaly (standard deviation of 0.2 degrees or less). The coplanar bias is appropriate for an object that is transferring from a parking orbit to a higher final orbit. The search is designed to cover the area between the two orbits where the target is most likely to be.

Like the previous search, the generated test script consists of a set of several sensor looks, and the test is considered a success if the target passes through the sensor field of view at least once during the execution of the test script.

5.2.4 Debris Re-Acquisition

The final tests of N-STASOS were piggybacked on ETS Debris Measurement Program sessions. During these sessions, new pieces of debris are sought by pointing a sensor at a particular part of the sky (called a stare field), watching, and waiting. If an object is seen passing through the field of view, the operator maneuvers the telescope to chase the target, acquire it, and establish a track on it. This track is used to generate an initial element set which is then used to re-acquire the object on a later orbit.

For this test, the initial element set is used to generate a search schedule, with uncertainties in the semi-major axis, the eccentricity, and the argument of perigee of the orbit. The eccentricity uncertainty will not affect the re-acquisition attempt because it occurs in approximately the same portion of the orbit as the initial track. In addition, if the target is to be re-acquired only one or two orbits later (as is usually the case), then the change of the perigee of the orbit will be relatively small, so the uncertainty in argument of perigee can be neglected [Pearce, 1993]. Therefore, searches generated to re-acquire debris pieces use only the uncertainty in the semi-major axis. The standard deviation of that uncertainty varies with the type of target.
As with the other searches, a test script is generated with several sensor looks, and the test is successful if the desired object passes through the field of view of the sensor at least once during the observing sequence.

Opportunities to attempt re-acquisition of a new debris piece are very limited, because the number of new pieces found varies widely from session to session, and there is no guarantee that any new debris will be located at all. Over the two nights of debris sessions in which the author participated, three new debris pieces were located.

5.3 Test Execution

Tests were run for a few hours each night between May 7th and May 10th. The following table shows the tests that were run. In Table 5-2, MDT is Mountain Daylight Time, UTC is Universal Time Coordinated, and the day is the day of the year (i.e. day 127 of 1993 is May 7th, 1993).
<table>
<thead>
<tr>
<th>Test No.</th>
<th>MDT Day:Time</th>
<th>UTC Day:Time</th>
<th>Object No.</th>
<th>Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>127:2100</td>
<td>128:0300</td>
<td>20962</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>2</td>
<td>127:2105</td>
<td>128:0305</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>3</td>
<td>127:2120</td>
<td>128:0320</td>
<td>22205</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>4</td>
<td>127:2125</td>
<td>128:0325</td>
<td>15235</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>5</td>
<td>127:2140</td>
<td>128:0340</td>
<td>22314</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>6</td>
<td>127:2145</td>
<td>128:0345</td>
<td>19713</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>7</td>
<td>127:2202</td>
<td>128:0402</td>
<td>22037</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>8</td>
<td>127:2210</td>
<td>128:0410</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>9</td>
<td>127:2225</td>
<td>128:0425</td>
<td>15237</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>10</td>
<td>127:2240</td>
<td>128:0440</td>
<td>20962</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>11</td>
<td>127:2245</td>
<td>128:0445</td>
<td>22195</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>12</td>
<td>127:2250</td>
<td>128:0450</td>
<td>22195</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>13</td>
<td>128:2010</td>
<td>129:0210</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>14</td>
<td>128:2055</td>
<td>129:0255</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>15</td>
<td>128:2100</td>
<td>129:0300</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>16</td>
<td>128:2105</td>
<td>129:0305</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>17</td>
<td>128:2110</td>
<td>129:0310</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>18</td>
<td>128:2125</td>
<td>129:0325</td>
<td>19751</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>19</td>
<td>129:1800</td>
<td>129:2400</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>20</td>
<td>129:1805</td>
<td>130:0005</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>21</td>
<td>129:1820</td>
<td>130:0020</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>22</td>
<td>129:1835</td>
<td>130:0035</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>23</td>
<td>129:2210</td>
<td>130:0410</td>
<td>20962</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>24</td>
<td>129:2215</td>
<td>130:0415</td>
<td>15235</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>25</td>
<td>129:2230</td>
<td>130:0430</td>
<td>22314</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>26</td>
<td>129:2235</td>
<td>130:0435</td>
<td>15237</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>27</td>
<td>129:2250</td>
<td>130:0450</td>
<td>8585</td>
<td>Geosynchronous</td>
</tr>
<tr>
<td>28</td>
<td>129:2255</td>
<td>130:0455</td>
<td>19713</td>
<td>Semi-synchronous</td>
</tr>
<tr>
<td>29</td>
<td>129:2305</td>
<td>130:0505</td>
<td>22195</td>
<td>Low Altitude</td>
</tr>
<tr>
<td>30</td>
<td>130:2125</td>
<td>131:0325</td>
<td>97121</td>
<td>Low Altitude Debris</td>
</tr>
<tr>
<td>31</td>
<td>131:0025</td>
<td>131:0625</td>
<td>97121</td>
<td>Low Altitude Debris</td>
</tr>
</tbody>
</table>

Table 5-2: N-STASOS Test Cases
5.3.1 May 7th Testing

On May 7th, 12 tests were planned. Seven geosynchronous test objects were chosen: one with no bias, three with a mean anomaly bias, and three with a semi-major axis bias. In addition, two semi-synchronous objects were chosen: one with an inclination bias and one with a coplanar bias. Finally, three low altitude objects were selected: one with no bias, and two with a bias in the right ascension of the ascending node.

Scheduling scripts for these tests were generated on a Silicon Graphics Indigo workstation during the day of May 7th, for execution that evening at the times specified in Table 5-2. These scripts were then transferred to the computer controlling the ETS-A dome. During the evening session, each of the test scripts was executed at the appropriate times. The telescope operator then watched the sensor field of view to look for each target.

Skies were cloudy during the session, so it was necessary to use the ETS satellite correlation software to identify where the desired object was relative to the position of the sensor for each sensor look. The correlation software uses the best known element set for the target. The object was considered located if and when it was predicted to have appeared in the telescope's field of view according to the computer display.

5.3.2 May 8th Testing

On May 8th, six tests were planned. All tests used the same semi-synchronous object. The first test was a repeat of an unsuccessful May 7th test, and the remaining
tests repeated the same search with an inclination bias for values of N varying from zero to four. As for the previous session, the test scripts were generated earlier in the day and transferred to the ETS-A dome computer,

Skies were overcast when the observing session began, so the ETS mount simulator was used instead of actually moving the telescope. The simulator contains a maneuver model for the telescope mount, and keeps track of all pointing constraints. The satellite correlation software was used to determine if and when the desired object passed through the sensor field of view.

5.3.3 May 9th Testing

Eleven tests were planned for May 9th. The first four were mean anomaly searches for a geosynchronous object, for N varying from zero to three, intended to be run by simulator before the evening session. The remaining seven tests included five geosynchronous objects (one with no bias, two with mean anomaly bias, and two with semi-major axis bias), a semi-synchronous object with a coplanar bias, and a low altitude object with no bias.

Scripts were generated earlier in the day and transferred to the ETS-A dome computer. The skies were clear that evening, so visual confirmation was used (i.e. targets could be seen on a television display of the telescope's field of view), with the satellite correlation program as a backup if the desired target was not well illuminated.
5.3.4 May 10th Testing

Two debris re-acquisition tests were run on May 10th. The first was generated earlier in the day using the initial element set for a UCT found during the debris observing session on May 9th. This script was transferred to the ETS-A dome computer and run later that evening.

A second script was created that evening to re-acquire the target on the next orbit. This script was generated approximately ten minutes before its scheduled execution. It was transferred to the ETS computer and executed on time.

5.4 Test Results

The results of the tests are shown in Table 5-3. In the table, N is the number of steps ahead used by N-STASOS to generate the search schedule. The results column contains the number of the look(s) during which the target passed through the sensor field of view. The last column shows the method of confirmation. If the target was not well illuminated, or obscured by clouds, the ETS satellite correlation software was used to determine if the object was in the field of view. As mentioned in the previous section, several tests were computer simulated without using the actual telescope when the weather was inclement on May 8th. As shown in Table 5-3, 25 of the 31 tests were successful.
<table>
<thead>
<tr>
<th>Test No.</th>
<th>Bias Type</th>
<th>N</th>
<th>Looks</th>
<th>Results</th>
<th>Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Visual</td>
</tr>
<tr>
<td>2</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>5, 8</td>
<td>Computer</td>
</tr>
<tr>
<td>3</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>4</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>9, 21</td>
<td>Computer</td>
</tr>
<tr>
<td>5</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>6</td>
<td>Coplanar</td>
<td>3</td>
<td>15</td>
<td>2, 9</td>
<td>Computer</td>
</tr>
<tr>
<td>7</td>
<td>RA of Asc. Node</td>
<td>3</td>
<td>15</td>
<td>Not Found</td>
<td>Computer</td>
</tr>
<tr>
<td>8</td>
<td>Inclination</td>
<td>3</td>
<td>30</td>
<td>Not Found</td>
<td>Computer</td>
</tr>
<tr>
<td>9</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>5, 19</td>
<td>Computer</td>
</tr>
<tr>
<td>10</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>11</td>
<td>None</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>12</td>
<td>RA of Asc. Node</td>
<td>3</td>
<td>24</td>
<td>2</td>
<td>Computer</td>
</tr>
<tr>
<td>13</td>
<td>Inclination</td>
<td>3</td>
<td>30</td>
<td>1, 2</td>
<td>Simulation</td>
</tr>
<tr>
<td>14</td>
<td>Inclination</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>Simulation</td>
</tr>
<tr>
<td>15</td>
<td>Inclination</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>Simulation</td>
</tr>
<tr>
<td>16</td>
<td>Inclination</td>
<td>2</td>
<td>10</td>
<td>1, 9</td>
<td>Simulation</td>
</tr>
<tr>
<td>17</td>
<td>Inclination</td>
<td>3</td>
<td>30</td>
<td>1</td>
<td>Simulation</td>
</tr>
<tr>
<td>18</td>
<td>Inclination</td>
<td>4</td>
<td>30</td>
<td>1, 2</td>
<td>Simulation</td>
</tr>
<tr>
<td>19</td>
<td>Mean Anomaly</td>
<td>0</td>
<td>12</td>
<td>Not Found</td>
<td>Simulation</td>
</tr>
<tr>
<td>20</td>
<td>Mean Anomaly</td>
<td>1</td>
<td>30</td>
<td>4, 17</td>
<td>Simulation</td>
</tr>
<tr>
<td>21</td>
<td>Mean Anomaly</td>
<td>2</td>
<td>30</td>
<td>4, 17</td>
<td>Simulation</td>
</tr>
<tr>
<td>22</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>3, 15</td>
<td>Simulation</td>
</tr>
<tr>
<td>23</td>
<td>None</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Visual</td>
</tr>
<tr>
<td>24</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>9, 28</td>
<td>Visual</td>
</tr>
<tr>
<td>25</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>1</td>
<td>Late</td>
<td>Visual</td>
</tr>
<tr>
<td>26</td>
<td>Mean Anomaly</td>
<td>3</td>
<td>30</td>
<td>5, 19</td>
<td>Visual</td>
</tr>
<tr>
<td>27</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Computer</td>
</tr>
<tr>
<td>28</td>
<td>Coplanar</td>
<td>3</td>
<td>15</td>
<td>3, 9</td>
<td>Visual</td>
</tr>
<tr>
<td>29</td>
<td>None</td>
<td>0</td>
<td>1</td>
<td>Late</td>
<td>Visual</td>
</tr>
<tr>
<td>30</td>
<td>Semi-Major Axis</td>
<td>3</td>
<td>20</td>
<td>5, 8, 9</td>
<td>Computer</td>
</tr>
<tr>
<td>31</td>
<td>Semi-Major Axis</td>
<td>2</td>
<td>20</td>
<td>Not Found</td>
<td>Visual</td>
</tr>
</tbody>
</table>

Table 5-3: N-STASOS Test Results
5.4.1 Acquisition Failures

The failure to acquire the low altitude target in test 7 was due to a typographical error in the element set. In test 8, the schedule script was executed approximately 15 seconds late due to operator error. The delay was large enough to prevent target acquisition. The same object was sought in test 13 and was successfully located. The failure to acquire these two targets is not the fault of the N-STASOS algorithm.

Test 25 produced a single look schedule because the uncertainty space was small enough to fit in one field of view for the sensor. Test 29 produced a single look schedule, as well, because it was a zero uncertainty search. When these test cases failed to acquire the desired objects, the telescope continued to stare at its scheduled pointing location. In each case, the target passed through five to 20 seconds later. As a result of using older element sets for these cases, the targets were outside the generated search spaces (i.e. the actual objects were more than three standard deviations away from the center of the search space).

5.4.2 N Variation Tests

Tests 14-18 repeated the same search for varying values of N, as did tests 19-22. Tests 14-18 each located the target. However, with N > 0, the object was found one look earlier than for the case with N = 0. Test 19, also with N = 0, did not acquire the target at all. However, tests 20-22 (using N > 0) did locate the desired object. These results initially suggested that using a look ahead strategy can produce a more efficient schedule than a local strategy (i.e. with N = 0). In reality, these results actually demonstrated the failure of the implementation of N-STASOS when N = 0.
In all cases, using a value of N greater than zero produced schedules which successfully located the desired object. However, later analysis of the schedule data showed that increasing N did not increase the overall likelihood of locating the target, nor did it improve the efficiency of the search in terms of sensor resource costs. This fact was the first evidence of errors present in the N-STASOS code. These errors are discussed further in Chapter 6.

5.4.3 Debris Re-Acquisition Tests

Tests 30 and 31 were the most risky tests, as they attempted to re-acquire a piece of debris located on May 9th. For test 30, a script was generated to locate the target from the previous night's element set. The object was located according to the ETS computer, but could not be seen on the display screen. Another re-acquisition attempt (test 31) was planned for the next orbit to determine if the object was not illuminated well enough to be seen, or if it was simply not at the location indicated by the computer. When the target was not found in test 31, it was determined that the uncertainty in semi-major axis (with a standard deviation of 50 kilometers) for that search was not chosen to be large enough by the author. The sensor was consistently behind the target throughout the search.
Chapter 6: Analysis

This chapter discusses analysis performed to determine the benefit of using an N-step look ahead scheduling strategy compared to a local strategy. The first section discusses errors that were found in the N-STASOS code after the completion of the field tests discussed in Chapter 5. The second section presents the results of tests run at Lincoln Laboratory to confirm that these errors were repaired and to establish the benefits of the N-step look ahead. The third section discusses the computational resources necessary to run N-STASOS, and the final section summarizes the analysis of N-STASOS.

6.1 N-STASOS Code Errors

After the field tests at ETS were complete, the results of several of the tests were verified under simulation at Lincoln Laboratory. The simulations showed that searches run with \( N = 0 \) did not run properly. In addition, search cases with \( N = 1, N = 2, \) and \( N = 3 \) produced schedules with identical overall values.

Examination of the N-STASOS code revealed errors which caused N-STASOS to perform bookkeeping on the probability space incorrectly. As a result, for the \( N = 0 \) cases, the schedules looked at the same part of the probability distribution for every look in the schedule. Figure 6-1 shows an \( N = 0 \) search for a geosynchronous object with a three degree uncertainty in mean anomaly (test 19 from Chapter 5). As shown in the figure, the \( N = 0 \) schedule would never acquire the target if it did not locate it on the first look. In this case, the sensor would constantly look behind the actual location of the satellite.
In Figure 6-1, the numbers on top of the boxes indicate the pointing sequence of the telescope, and the number on the bottom correspond to the position of the satellite for each look in the sequence.

Further tests were run for other values of N. N-STASOS produced valid schedules for these test cases (i.e. they could be used to locate the desired target), but they did not gain the benefit of an N-step look ahead. Test cases were simulated searching for the same geosynchronous object, at the same time, but with values of N varying from one to three. The schedules produced were identical to one another. Examination of the N-STASOS code showed that the program did not properly account for previously viewed probability space when considering the N steps ahead. Changes in the probability space occurred only for the current look calculated by the program. Therefore, the program would always choose the local schedule as the best schedule for any value of N. In other words, the scheduler that was successfully tested at ETS actually used a local strategy.
6.2 Benefits of the N-Step Look Ahead

The errors in the N-STASOS code were corrected, and additional tests were run at Lincoln Laboratory to determine what, if any, benefit could be gained by using an N-step look ahead over what was demonstrated to be a successful local scheduler. The example shown here depicts a search for a geosynchronous object with a five degree standard deviation in uncertainty on the mean anomaly of the element set. Search cases were simulated for both a ground-based sensor (the ETS-A dome telescope) and a space-based sensor (the SBV).

Like the test cases run at ETS, the ground-based search was 10 minutes long, and assumed a 20 second average look time. The space-based search was 20 minutes long, with an average look time of two minutes. These times differ because of the MSX's longer maneuver and settle times.

Figures 6-2 and 6-3 show the amount of probability space covered by each search and the value of each schedule, with the value function for these cases shown in Equation 6-1.

\[ f_{value} = 100P_{appear} - 0.1t_{maneuver} \]

(6-1)

Because of the structure of the value function, the increased maneuver times required by the MSX will naturally produce schedules with smaller values. The results shown in the figures are typical of the test cases simulated at Lincoln Laboratory after the implementation corrections were made.
In Figure 6-2, the solid line represents the probability of appearance of the target, and the dashed line is overall schedule value. While the total probability of finding the object during the search does not appreciably increase with N, the overall schedule value does. This means that the sensor's resources are used more efficiently for higher values of N. (In this case, the only resource considered is maneuver time.) For N = 2 and N = 4, the schedule value is approximately five percent higher than for N = 0 (78.4 compared to 74.4).
For the space-based search, there is a slight increase in total probability captured by the schedule (solid line in Figure 6-3) as N increases. The change in value (dashed line), however, is much more apparent than for the ground-based example. Although the value of the schedule drops substantially as N increases from zero to one, further increases to N = 2, N = 3, and N = 4 produce more efficient schedules. The value of the schedule where N = 4 is over 50 percent greater than for the N = 0 case (16.8 compared to 11.0).

The drop in schedule value experienced for N = 1 is not a regular occurrence. Occasionally, N-STASOS will determine that for a particular look, it is better to choose an area with a smaller value, believing that such a choice will provide a higher-valued opportunity on the next look. For N = 1, the scheduler can do this several times in succession and, because of its limited look ahead, come to the end of the schedule before the higher-valued opportunity is finally chosen. Therefore, N-STASOS can produce a
less efficient schedule. This problem is corrected by increasing the number of steps looked ahead by the scheduler.

Additional similar search simulations performed at Lincoln Laboratory confirm the general trend shown here. The tests suggest that using an N-step look ahead strategy can produce a schedule that is more efficient than a local strategy might generate. However, the modest gains experienced for ground-based sensors may not justify its use given the increase in computational resources necessary.

For space-based sensors, the use of an N-step look ahead can help to conserve valuable spacecraft resources, and so should be a valuable tool for scheduling search tasks for sensors like the SBV.

6.3 Computational Resources

During the field testing session at ETS, the amount of central processing unit (CPU) time used by N-STASOS was measured for a ten minute, 30 look, mean anomaly search with a three degree standard deviation on its uncertainty. The results of this test are shown in Figures 6-4 and 6-5.
Figure 6-4: N-STASOS CPU Usage (N = 0 to N = 3)

Figure 6-5: N-STASOS CPU Usage (N = 0 to N = 5)

For values of N up to three, the CPU usage time is approximately a linear function of N, with a slope of approximately 20 seconds per look ahead, and an intercept of
approximately 150 seconds. However, as shown in Figure 6-5, the CPU usage increases more sharply for \( N = 4 \) and \( N = 5 \). This sharp increase indicates that the N-STASOS program has utilized all available memory (16 megabytes on the Silicon Graphics Indigo workstation), and is using the workstation’s hard drive for additional swap space. The use of the hard drive swap file is a much slower process than normal on-board memory access. Because of these workstation limitations, N-STASOS tests were limited to values of \( N \) of four or less.

6.4 Analysis Summary

The field tests at ETS did not actually demonstrate the validity of an \( N \)-step look ahead scheduling strategy. Due to errors in the N-STASOS code that were not detected before field testing began, a local scheduler was actually tested. While these tests successfully demonstrated that the local scheduler worked well, they did not provide any information that would help to make any conclusions about the advantages gained using an \( N \)-step look ahead.

To remedy the situation, the code was corrected and several test cases were run under simulation at Lincoln Laboratory. These test cases showed that using a look ahead scheduling strategy produces searches that make use of sensor resources more efficiently than locally strategies. For one particular space-based search, using \( N = 4 \) produced a search schedule whose overall value was more than 50 percent greater than a schedule generated using \( N = 0 \).
Chapter 7:  
Conclusions

7.1 Summary

The N-Step Ahead Single Object Scheduler (N-STASOS) is a software package that has been developed to efficiently schedule space object searches from a space-based, optical, angles-only sensor like Lincoln Laboratory's Space Based Visible (SBV) sensor.

N-STASOS uses an N-step look ahead strategy which allows it to minimize missed scheduling opportunities that may occur when using a local strategy, but does so at a lower computational cost than a global scheduling strategy.

N-STASOS was written in the C programming language on a Silicon Graphics Indigo workstation, and complies with the ANSI C standard, so that it can be ported to multiple platforms with minimal changes to the code.

Computer simulations were performed at Lincoln Laboratory. These simulations indicated that using an N-step look ahead strategy produces can search schedules which make more efficient use of sensor resources. This efficiency is particularly noticeable for space-based sensors, where resource usage is as much as 50 percent less than for a local schedule. Because resource conservation is a lower priority for ground-based sensors, gains in ground-based schedule efficiency are very small when the N-step look ahead strategy is used (on the order of five percent). It is probably sufficient to use a local strategy in these cases.
Field tests were carried out at the Experimental Test Site (ETS), a ground-based sensor site near Socorro, New Mexico. Although errors in the implementation of N-STASOS negated the advantages of the N-step look ahead, the scheduler was still able to find its target 25 times out of 31 attempts (80.6 percent), showing that it is a valid method of scheduling space object searches, at least using a local scheduling strategy. (It should be noted that two of the five unsuccessful attempts were due to poor execution or to the use of a bad element set, and do not indicate an N-STASOS failure.)

This project has shown that an N-step look ahead strategy can be a valuable technique for scheduling space object searches from a space-based optical sensor. N-STASOS should make a valuable addition to the suite of software packages that have been developed for use by the SBV sensor.

7.2 Recommendations for Further Work

This section presents a few ideas for improvements and extensions which can be made to N-STASOS to make it run more efficiently, and make it work better with sensors like the SBV.

7.2.1 Detectability Model

Time constraints did not allow the inclusion of a detectability model in the current version of N-STASOS. Such a model would indicate the brightness of the desired object. Because no detectability model was included, there were test cases where N-STASOS produced a schedule that correctly located the object, but that object could not be seen due to insufficient illumination. Having a model of the illumination of the target would allow N-STASOS to operate more efficiently.
7.2.2 Use of Multiple CCDs

The SBV sensor has four CCDs on its focal plane, only one of which can be used at a time. The current implementation of N-STASOS assumes that only CCD #3 is used, because it is aligned with the MSX boresite. It would be useful to augment the maneuver model to determine where it would be better to simply switch to a different CCD than to maneuver the whole satellite.

7.2.3 SBV Script Generation

A subroutine was written for N-STASOS that converts the generated schedule into a script that can be used by the sensors at ETS. However, no script generator was made for the SBV. The ability to generate SBV pointing commands will help to further automate N-STASOS.

7.2.4 Improved Constraint Models and Simulator Interface

Currently, N-STASOS is equipped with rudimentary constraint determination which only allows consideration of hard constraints affecting the boresite of the SBV/MSX. Constraint models should be improved to account for those constraints that affect other axes. Cost functions associated with soft constraints should also be included.

It would also be useful to interface N-STASOS with the SBV simulator currently operating at Lincoln Laboratory. Schedules generated by N-STASOS could be run through the simulator, which uses far more exhaustive models for pointing constraints.
7.2.5 Multiple Object Searches

While N-STASOS was designed primarily to search for single space objects, the algorithm should be applicable to multiple object searches with minimal changes. This would allow N-STASOS to generate catalog maintenance schedules for the SBV.

7.2.6 Improved Memory Usage

As discussed in Chapter 6, N-STASOS can overload the workstation on which it is run for larger values of N. It would be useful to examine the way N-STASOS uses the workstation’s on-board memory to see if its memory allocation can be better managed, or if it would be better to transfer the program to a more powerful platform.
References


Gibson, D.M. *Experimental Test System (ETS) Executive Summary.*, Private Communication.


Pearce, Eric C. Personal communication with author, 1993.


Taff, L.G. *Observing Artificial Satellites.*, Private Communication.


——— *Optical Artificial Satellite Searches.*, Private Communication.


* A Long Term GEODSS Search Plan, Private Communication.


Appendix A:
N-STASOS Flowcharts

This appendix provides two diagrams. The first is a top-level flowchart showing the sequence of N-STASOS subroutine and function calls. The second diagram shows the program flow for the inner loop that makes the actual scheduling choices during program execution. These diagrams appear on the following two pages.

Descriptions of each of the subroutines and functions in the diagrams can be found in Appendices B and C.
Figure A.1: N-STASOS Main Program Flowchart
Figure A.2: N-STASOS Inner Loop Flowchart
This page left intentionally blank
Appendix B: N-STASOS Subroutines

This appendix serves as a programmer's manual for the N-Step Ahead Single Object Scheduler. It provides a brief description of the function of each of the subroutines developed for N-STASOS, and shows the calling sequences necessary to execute them.

B.1 Subroutine adjust_bucket

The subroutine adjust_bucket adjusts the figures of merit for all entries in the bucket array to account for pseudo-objects that have been viewed. The calling sequence for this subroutine is as follows:

adjust_bucket(x, y, buck, map, plist, info, sign);

where:

\[ x, y \] = the integer coordinates of the bucket array record containing the RA/DEC position of the viewed objects, i.e. \[ \text{buck}[x][y] \].

\[ \text{buck} \] = the bucket array, which is altered by the subroutine.

\[ \text{map} \] = an array of records containing the RA/DEC positions of each pseudo-object.

\[ \text{plist} \] = a list of double precision probabilities of appearance of each pseudo-object.

\[ \text{info} \] = a structure containing user input to the program.

\[ \text{sign} \] = a double precision number indicating whether the figure of merit is to be augmented or reduced (= +1.0 for addition, -1.0 for subtraction).
B.2 Subroutine adjust_map

The subroutine adjust_map sets a flag which indicates that the given pseudo-objects have been viewed, and should no longer be considered. The calling sequence for this subroutine is as follows:

`adjust_map(map, objects, end, info, used);`

where:

- `map` = an array of records containing the RA/DEC positions of each pseudo-object. The array is altered by the subroutine.
- `objects` = a list of integer object ID numbers indicating the objects to be marked as viewed.
- `end` = a pointer to the end of the object list.
- `info` = a structure containing user input to the program.

B.3 Subroutine append_blist

The subroutine append_blist appends a list of bucket ID numbers to a pseudo-object's entry in the map array. The calling sequence for this subroutine is as follows:

`append_blist(map, frame, no, bucknum);`

where:

- `map` = an array of records containing the RA/DEC positions of each pseudo-object. The array is altered by the subroutine.
- `frame` = the frame number of the map array being altered
- `no` = the number of the map entry being altered, i.e. `map[frame][no]`
- `bucknum` = the bucket number to be appended to the map array entry.
B.4 Subroutine append_map

The subroutine **append_map** appends the next frame of map information to the end of the input map array. The calling sequence for this subroutine is as follows:

```plaintext
append_map(map, now, info);
```

where:

- **map** = an array of records containing the RA/DEC positions of each pseudo-object. The array is altered by the subroutine.
- **now** = an ORBLIB time structure containing the time for the next map frame.
- **info** = a structure containing user input to the program.

B.5 Subroutine append_olist

The subroutine **append_olist** appends a list of pseudo-object ID numbers to a bucket's entry in the bucket array. The calling sequence for this subroutine is as follows:

```plaintext
append_olist(buck, frame, no, objnum);
```

where:

- **buck** = an array of records containing the RA/DEC positions of each position considered by the algorithm. The array is altered by the subroutine.
- **frame** = the frame number of the bucket array being altered.
- **no** = the number of the bucket entry being altered, i.e. `buck[frame][no]`.
- **objnum** = the object number to be appended to the bucket array entry.
B.6 Function assign_bucket

The function `assign_bucket` creates an array of records containing the positions and associated figures of merit of all fields of view for the sensor considered by the algorithm. The calling sequence for this function is as follows:

```
bucket_array = assign_bucket(map, info, plist, *totbuck);
```

where:

<table>
<thead>
<tr>
<th>buck</th>
<th>= an array of records containing the RA/DEC positions of each position considered by the algorithm. The array is created by the subroutine.</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>= a structure containing user input to the program.</td>
</tr>
<tr>
<td>plist</td>
<td>= a list of double precision probabilities of appearance of each pseudo-object in the catalog.</td>
</tr>
<tr>
<td>*totbuck</td>
<td>= a double precision vector containing the number of records in each frame of the bucket array.</td>
</tr>
</tbody>
</table>

B.7 Function assign_cat

The function `assign_cat` generates a linked list of pseudo-object element sets with the associated probability of appearance of each. The calling sequence for this function is as follows:

```
catalog = assign_cat(mean, stdev, step, numsd, *numobj);
```

where:

<table>
<thead>
<tr>
<th>mean</th>
<th>= the mean element set, in ORLIB format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdev</td>
<td>= an ORLIB element set containing the uncertainties for each orbital element, expressed as a standard deviation.</td>
</tr>
<tr>
<td>step</td>
<td>= an ORLIB element set containing the granularity to use for each orbital element.</td>
</tr>
</tbody>
</table>
numsd = an integer indicating the number of standard deviations to be considered about the mean when creating the catalog.
*numobj = an integer with the number of objects in the catalog. This number is returned by the function.

B.8 Function assign_map

The function assign_map creates an array of records containing the positions of all input pseudo-objects over time. The calling sequence for this function is as follows:

map_array = assign_map(info);

where:

info = a structure containing user input to the program.

B.9 Function assign_prob

The function assign_prob creates an array containing the probability of appearance of each pseudo-object in the given catalog. The calling sequence for this function is as follows:

prob_list = assign_prob(catalog, numobj);

where:

catalog = a linked list of element sets from the pseudo-object catalog.
numobj = the number of element sets in the catalog list.
B.10 Function assign_value

The function assign_value determines the figure of merit for each bucket in the bucket array, and assigns that value to the array entry. The calling sequence for this function is as follows:

\[
\text{value} = \text{assign\_value}(\text{map}, \text{buck}, \text{plist}, \text{num}, \text{ra}, \text{dec}, \text{frame}, \text{nobj}, \text{gors}, \text{nunc});
\]

where:

- \text{map} = the map array with all pseudo-object positions.
- \text{buck} = the bucket array with all considered sensor pointings.
- \text{plist} = a double precision list of probabilities of appearance of each pseudo-object.
- \text{num} = an integer bucket ID number, to which the value is assigned.
- \text{ra, dec} = the RA/DEC position of the center of the bucket (double precision).
- \text{frame} = an integer number with the bucket and map frame to be affected.
- \text{nobj} = an integer indicating the number of pseudo-objects in each frame of the map array.
- \text{gors} = an integer (character) that indicates whether the sensor site is (G)round-based or (S)pace-based.
- \text{nunc} = an integer with the number of uncertain orbital elements.

B.11 Function bound

The function bound bounds a given angle between zero and 360 degrees. The calling sequence for this function is as follows:

\[
\text{bounded\_angle} = \text{bound}(\text{num});
\]

where:
num = angle to be bounded.

B.12 Subroutine cartosper

The subroutine cartosper takes a three-dimensional Cartesian vector and converts it to Right Ascension / Declination coordinates. The calling sequence for this subroutine is as follows:

```
cartosper(*invect, *outvect);
```

where:

- *invect = a 3-dimensional Cartesian vector to be converted to RA/DEC.
- *outvect = a 2-dimensional vector containing the RA/DEC output from the subroutine.

B.13 Function constr

The function constr determines if hard pointing constraints have been violated for a given sensor pointing. It returns a one if a violation occurs, and zero otherwise. The calling sequence for this subroutine is as follows:

```
constraint_flag = constr(*sitevec, ra, dec, epoch, gors);
```

where:

- *sitevec = a 3-dimensional ECI vector pointing to the sensor site at the given time.
- ra, dec = the (double precision) RA/DEC pointing to be examined for constraint violations.
- gors = an integer (character) that indicates whether the sensor site is
(G)round-based or (S)pace-based.

B.14 Subroutine create_map_frame

The subroutine create_map_frame creates one frame of an array of records containing the positions of all pseudo-objects in the input catalog. The calling sequence for this subroutine is as follows:

```
create_map_frame(map, frame, now, info);
```

where:

- `map` = the map array to be appended by this subroutine.
- `frame` = the number (integer) of the frame of the map array to be created.
- `now` = the time in ORBLIB format.
- `info` = a structure containing user input to the program.

B.15 Subroutine elset_to_radec

The subroutine `elset_to_radec` takes an element set for a given pseudo-object and converts it to site-based Right Ascension / Declination coordinates for a given time. The calling sequence for this subroutine is as follows:

```
elset_to_radec(satset, sitepos[], radec[], epoch, new, bstar);
```

where:

- `satset` = an ORBLIB element set for the object to be converted.
- `sitepos[]` = a 3-dimensional ECI vector for the sensor site.
- `radec[]` = a 2-dimensional vector with the RA/DEC of the object
new = an integer parameter used by ORBLIB.
bstar = the double precision B-star coefficient used by the SGP4 propagator.

B.16 Subroutine find_best

The subroutine find_best searches the bucket array to determine which N-step schedule has the highest figure of merit. It returns the first step of that schedule. The calling sequence for this subroutine is as follows:

\[
\text{find_best}(\text{buck, map, plist, *totbuck, info, *xret, *yret, *valret});
\]

where:

- \text{buck} = the bucket array to be searched.
- \text{map} = the map array containing all pseudo-object positions.
- \text{plist} = the list containing the probabilities of appearance of all objects.
- \text{*totbuck} = an integer list containing the number of records in each frame of the bucket array.
- \text{info} = a structure containing user input to the program.
- \text{*xret} = an integer with the x-coordinate of the best bucket (returned).
- \text{*yret} = an integer with the y-coordinate of the best bucket, i.e. \text{buck[xret][yret]} (returned).
- \text{*valret} = the figure of merit of the best bucket (double precision, returned).

B.17 Subroutine fine_tune

The subroutine fine_tune fine tunes the coarse schedule generated by N-STASOS by determining how long the sensor should remain at each position in the schedule. The calling sequence for this subroutine is as follows:

\[
\text{fine_tune(*schedule, info, plist)};
\]
*schedule* = a schedule list produced by N-STASOS. This list is altered by the subroutine.

*info* = a structure containing user input to the program.

*plist* = a list containing the double precision probabilities of appearance of each pseudo-object in the catalog.

**B.18 Function get_cat**

The function *get_cat* retrieves a list of pseudo-object element sets from a given file. The calling sequence for this function is as follows:

```c
catalog = get_cat(*file, *numunc, *numobj, *bstar);
```

where:

*file* = a string containing the input file name.

*numunc* = an integer containing the number of uncertain elements (returned).

*numobj* = an integer containing the number of element sets in the catalog (returned).

*bstar* = the double precision SGP4 B-star coefficient (returned).

**B.19 Subroutine get_corners**

The subroutine *get_corners* determines the maximum and minimum Right Ascension and Declination for a given frame of the map array. The calling sequence for this subroutine is as follows:

```c
get_corners(map, frame, size, *minra, *maxra, *mindc, *maxdc);
```
where:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>map</code></td>
<td>the map array to be searched.</td>
</tr>
<tr>
<td><code>frame</code></td>
<td>the frame of the map array to be searched (integer).</td>
</tr>
<tr>
<td><code>size</code></td>
<td>the number of pseudo-objects in the map array (integer).</td>
</tr>
<tr>
<td><code>*minra</code></td>
<td>the minimum value of right ascension (double precision, returned).</td>
</tr>
<tr>
<td><code>*maxra</code></td>
<td>the maximum value of right ascension (double precision, returned).</td>
</tr>
<tr>
<td><code>*mindc</code></td>
<td>the minimum value of declination (double precision, returned).</td>
</tr>
<tr>
<td><code>*maxdc</code></td>
<td>the maximum value of declination (double precision, returned).</td>
</tr>
</tbody>
</table>

**B.20 Function get_elset**

The function `get_elset` reads an element set in standard ORBLIB format, plus the B-star coefficient used by the SGP4 propagator, from the specified file. The calling sequence for this function is as follows:

```
element_set = get_elset(*file, *bstar);
```

where:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*file</code></td>
<td>a string containing the name of the input file.</td>
</tr>
<tr>
<td><code>*bstar</code></td>
<td>the double precision SGP4 B-star coefficient, read from the file and returned with the element set.</td>
</tr>
</tbody>
</table>

**B.21 Function get_time**

The function `get_time` converts the given time in year/month/day/etc. or year/day/etc. to ORBLIB format (days to epoch J2000). The calling sequence for this function is as follows:

```
ORBLIB_time = get_time();
```
B.22 Function input

The function **input** gathers all user input for N-STASOS, including: the pseudo-object element set catalog, start time and end time for the search, average look time, initial RA/DEC position of sensor, and N, the number of steps to look ahead. The calling sequence for this function is as follows:

```c
input_information = input(*filename);
```

where:

*filename = a string containing the name of the input file.

B.23 Function input_elset

The function **input_elset** reads in an element set entered by the user, in standard ORBLIB format. The calling sequence for this function is as follows:

```c
element_set = input_elset(time, *bstar);
```

where:

<table>
<thead>
<tr>
<th>time</th>
<th>an integer which is 1 if the time is to be inputted, 0 otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>*bstar</td>
<td>the double precision B-star coefficient used by the SGP4 propagator.</td>
</tr>
</tbody>
</table>
B.24 Function man_time

The function **man_time** determines the amount of time necessary to maneuver between two points. It contains separate maneuver models for ground-based and space-based sensors. The calling sequence for this function is as follows:

\[
\text{maneuver\_time} = \text{man\_time}(ra1, dc1, ra2, dc2, gors);
\]

where:

\[
\begin{align*}
ra1, dc1 &= \text{the initial RA/DEC position (double precision).} \\
ra2, dc2 &= \text{the final RA/DEC position (double precision).} \\
gors &= \text{an integer (character) indicating whether the site is (G)round-based or (S)pace-based.}
\end{align*}
\]

B.25 Function nprob

The function **nprob** determines the normal probability of the value of a variable given the mean and standard deviation of its distribution. The calling sequence for this function is as follows:

\[
\text{probability} = \text{nprob}(x, \text{mean}, \text{stdev});
\]

where:

\[
\begin{align*}
x &= \text{the position on the distribution at which the probability is sought (double precision).} \\
\text{mean} &= \text{the mean of the distribution (double precision).} \\
\text{stdev} &= \text{the standard deviation of the distribution (double precision).}
\end{align*}
\]
B.26 Function parents

The function parents finds the array index of the parent nodes of the input tree node. The calling sequence for this function is as follows:

\[
\text{parent\_node\_list} = \text{parents}(\text{level}, \text{node});
\]

where:

\[
\begin{align*}
\text{level} & = \text{the level of the input node in the tree (integer).} \\
\text{node} & = \text{the number of the input node (integer).}
\end{align*}
\]

B.27 Function real_degrees

The function real_degrees determines the number of degrees of Right Ascension that are equivalent to one degree of "real" space for a given value of Declination. The calling sequence for this function is as follows:

\[
\text{RA\_degrees\_per\_degree} = \text{real\_degrees}(\text{dec});
\]

where:

\[
\text{dec} = \text{the declination at which the function is calculated (double precision).}
\]

B.28 Subroutine write_bucket

The subroutine write_bucket writes the bucket array to a file. The calling sequence for this subroutine is as follows:
write_bucket(buck, *totbuck, frames);

where:

buck = the bucket array to be written to a file.
*totbuck = an integer list of the number of records in each array frame.
frames = an integer containing the number of frames to be written.

B.29 Subroutine write_cat

The subroutine write_cat writes the generated element set catalog to a file. The calling sequence for this subroutine is as follows:

write_cat(*file, catalog, bstar);

where:

*file = a string containing the name of the file to be written.
catalog = a list containing the pseudo-object catalog.
bstar = the double precision B-star coefficient used by the SGP4 propagator.

B.30 Subroutine write_ets_script

The subroutine write_ets_script takes the schedule output from N-STASOS and converts it into a script useable by the telescope controllers at ETS. The calling sequence for this subroutine is as follows:

write_ets_script(*schedule, ntotlooks, tavg);
where:

\*schedule = a list containing the search schedule generated by N-STASOS.
ntotlooks = the number of looks in the schedule (integer).
tavg = the average time between looks (seconds, double precision).
Appendix C: ORBLIB Subroutines

This appendix provides a brief description of the function of each of the subroutines from the ORBLIB orbital mechanics library that are used by N-STASOS. ORBLIB is a operational FORTRAN library developed at Lincoln Laboratory [Lander, 1987]. The calling sequences shown are those necessary to call the FORTRAN subroutines from a C language program.

C.1 Subroutine JAMEOE

The subroutine JAMEOE propagates the given mean element set to the specified time and returns an osculating element set. The calling sequence for this subroutine is as follows:

```fortran
FortranName(jameoe)(&melset, &utd, &utf, &params,
         &new, &oelset, &error);
```

where:

- `&melset` = the address of the mean element set to be propagated (in ORBLIB format).
- `&utd, &utf` = the addresses of the time to which the element set is to be propagated in integral and fractional days to epoch J2000 (double precision).
- `&params` = the address of a double precision list of force model parameters.
- `&new` = the address of an integer which determines what initialization is to be performed by the subroutine.
- `&oelset` = the address of the osculating element set that is returned.
- `&error` = the address of the ORBLIB error reporting vector.
C.2 Subroutine JOEGIC

The subroutine JOEGIC calculates the geocentric equatorial inertial position and velocity of an object from its osculating element set. The calling sequence for this subroutine is as follows:

FortranName(joegic)(&elset, &position, &error);

where:

&elset = the address of the element set from which the position and velocity of the object are to be calculated.
&position = the address of a 3 x 2 double precision array containing the position and velocity of the object (returned).
&error = the address of the ORBLIB error reporting vector.

C.3 Subroutine JDSGRC

The subroutine JDSGRC takes position in geocentric rotating geodetic coordinates (latitude, longitude, and altitude), and returns geocentric rotating Cartesian coordinates. The calling sequence for this subroutine is as follows:

FortranName(jdsgrc)(&latitude, &longitude, &altitude, &position, &error);

where:

&latitude = the address of the double precision latitude (degrees).
&longitude = the address of the double precision longitude (degrees E).
&altitude = the address of the double precision altitude (meters).
&position = the address of the returned 3-vector with position (double precision).
&error = the address of the ORBLIB error reporting vector.
C.4 Subroutine JGRCIC

The subroutine JGRCIC takes the geocentric rotating coordinates returned by JDSGRC and converts them to geocentric inertial Cartesian coordinates. The calling sequence for this subroutine is as follows:

FortranName(JGRCIC)(&utd, &utf, &rpos, &ipos, &error);

where:

&utd, &utf = the addresses of the time at which the position is to be determined, in integral and fractional days to epoch J2000 (double precision).
&rpos = the address of a 3 x 2 array containing the rotating position and velocity (double precision).
&ipos = the address of a 3 x 2 array containing the inertial position and velocity (double precision, returned).
&error = the address of the ORBLIB error reporting vector.

C.5 Subroutine JDDAY

The subroutine JDDAY converts time from year, day, hour, minute, and second into integral and fractional days to epoch J2000. The calling sequence for this subroutine is as follows:

FortranName(jdday)(&year, &day, &hour, &minute, &second, &utd, &utf, &error);

where:

&year = the address of the number of years since 1900
C.6 Subroutine JMDAY

The subroutine JMDAY converts time from year, month, day, hour, minute, and second into integral and fractional days to epoch J2000. The calling sequence for this subroutine is as follows:

FortranName(jday)(&year, &month, &day, &hour, &minute,&second, &utd, &utf, &error);

where:

&year = the address of the number of years since 1900 (double precision).
&month = the address of the month of the year (double precision).
&day = the address of the day of the month (double precision).
&hour = the address of the hour (double precision).
&minute = the address of the minute (double precision).
&second = the address of the second (double precision).
&utd, &utf = the addresses of the time to be returned, in integral and fractional days to epoch J2000 (double precision).
&error = the address of the ORBLIB error reporting vector.