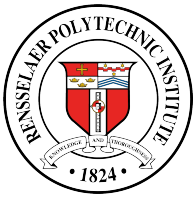
TRACKING OF SPACE DEBRIS FROM PUBLICLY AVAILABLE DATA

Philip Hoddinott

Submitted in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Approved by:
Kurt Anderson, Chair
John Christian
Matthew Oehlschlaeger



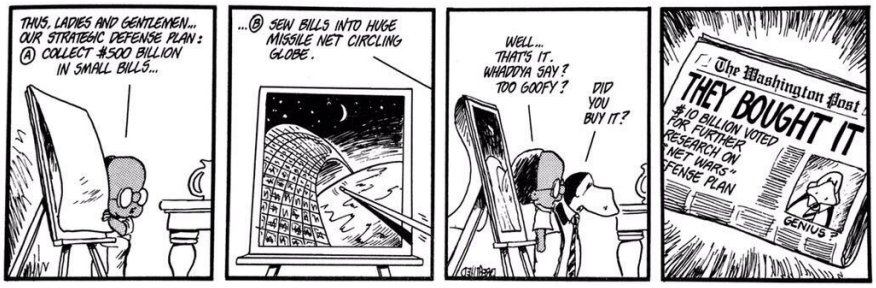
Department of Mechanical, Aerospace, and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York
November 2018

October 23, 2018

Contents

But, Captain, I cannot change the laws of physics

-Lt. Commander Scott (Scotty)
USS *Enterprise*



List of Tables

List of Figures

0.1 Acknowledgments

This paper would never have been possible with out the help of so many people. I am indebted to my parents and my brother for their support.

John Benke for his assitance with Github and PHP. I am thankfull to Paul McKee for his assistance with Latex. Thank lots of people here

Mom, dad, David, P Anderson John B for git and PHP Paul McKee for latex

0.2 Abstract

The purpose of this project is to acquire information about earth orbiting debris, turn the information into a usable format, and develop a targeting algorithm for a debris removal satellite. The information is things such as orbital elements, debris size, and sping if i can find htis.

Should this be accomplished in a timely manner more work will be done on the orbital dynamics of OSCAR getting to said debris.

0.3 Introduction

Before the methods of this project are explored, the background of the problem must first be presented. Space debris and the threat they pose will be explained, as well as the proposed solution.

0.3.1 Space Debris

March 17th, 1958 the Vanguard 1 satellite was launched. It was the fourth man made satellite. Despite communication being lost in the 1960s, it is still in orbit to this day[6].

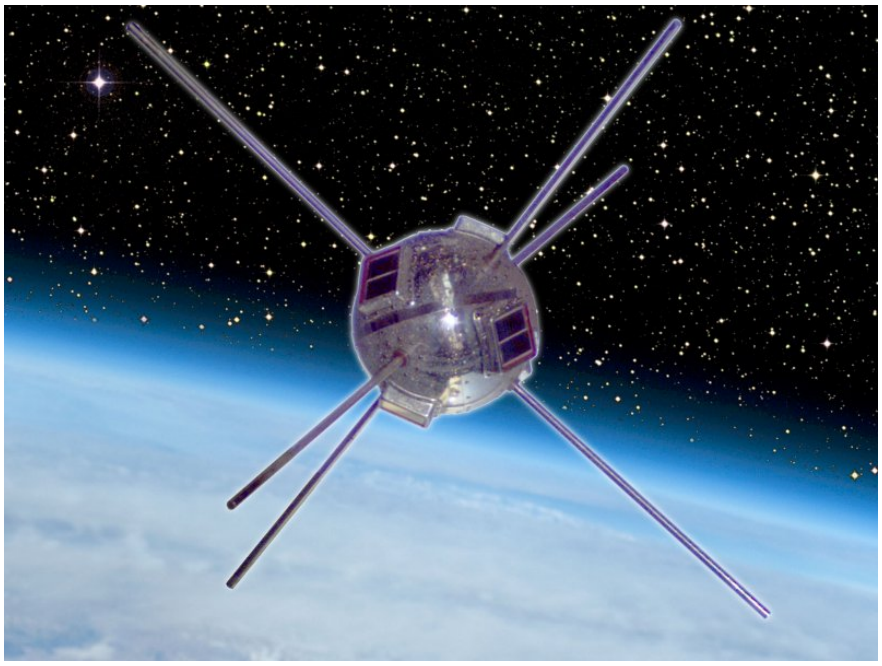


Figure 1: Artist's rendition of Vanguard 1 in orbit.

While spacecraft are supposed to be moved to a graveyard orbit, not all are. These large objects would be catastrophic if they collided with another object in orbit. However large objects do not pose the greatest threat. Their size makes them easy to track and there are comparatively little of them.

The debris that poses the greatest threat are the smaller pieces. Their size makes them harder to track and predict trajectories. And they are numerous. As of 2013 it is estimated that there are:

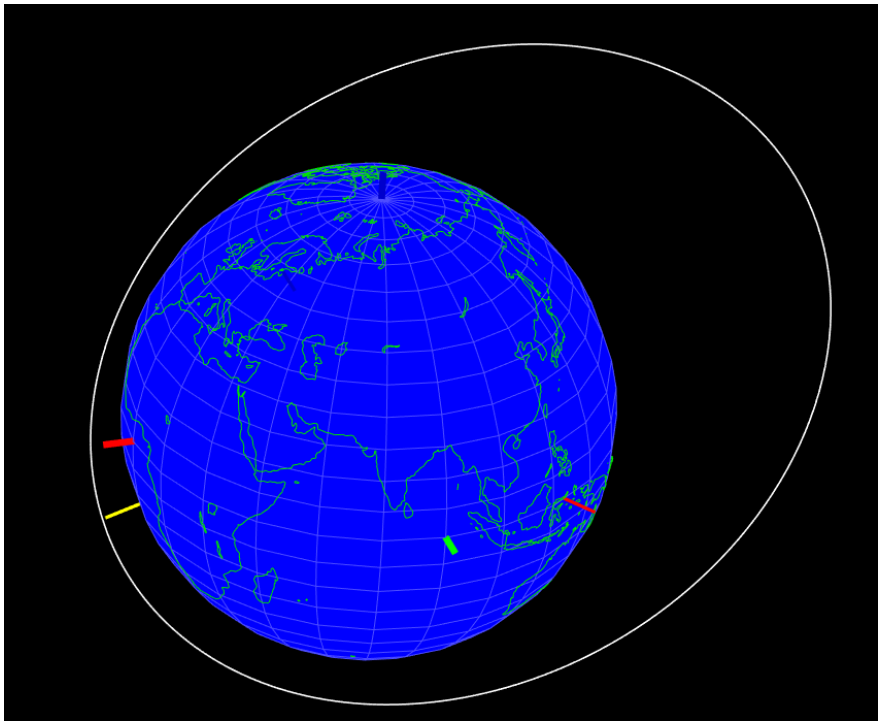


Figure 2: Vanguard's Orbit as of 10/17/2018 UPDATE THIS

29,000 Objects larger than 10 cm.

670,000 Objects larger than 1 cm.

Over 170,000,000 Objects larger than 1 mm.

These objects can do grievous harm to people or objects in space. A 10 cm object can cause the destruction of a satellite, a 1 cm object could penetrate the ISS's shields putting lives at risk and a 1 mm object could destroy critical subsystems.[7] The impact of one of these objects is shown in figure 3.

0.3.2 Kessler Syndrome

More than just harm to individual spacecraft or persons there is the threat of Kessler Syndrome. Also known as the Kessler effect or collision cascading, this is a scenario where there are enough objects in low earth orbit that a collision between objects will cause a chain re-

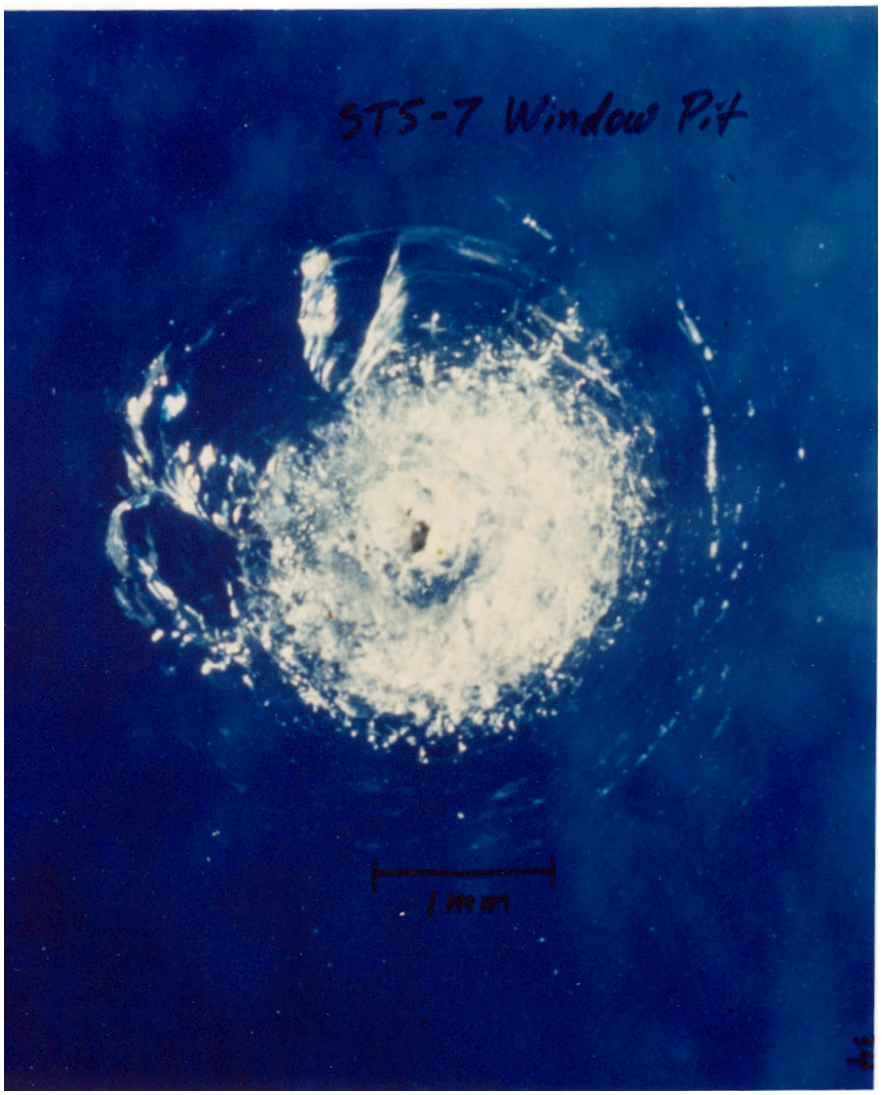


Figure 3: An impact crater on one of the windows of the Space Shuttle Challenger following a collision with a micrometeoroid[2]

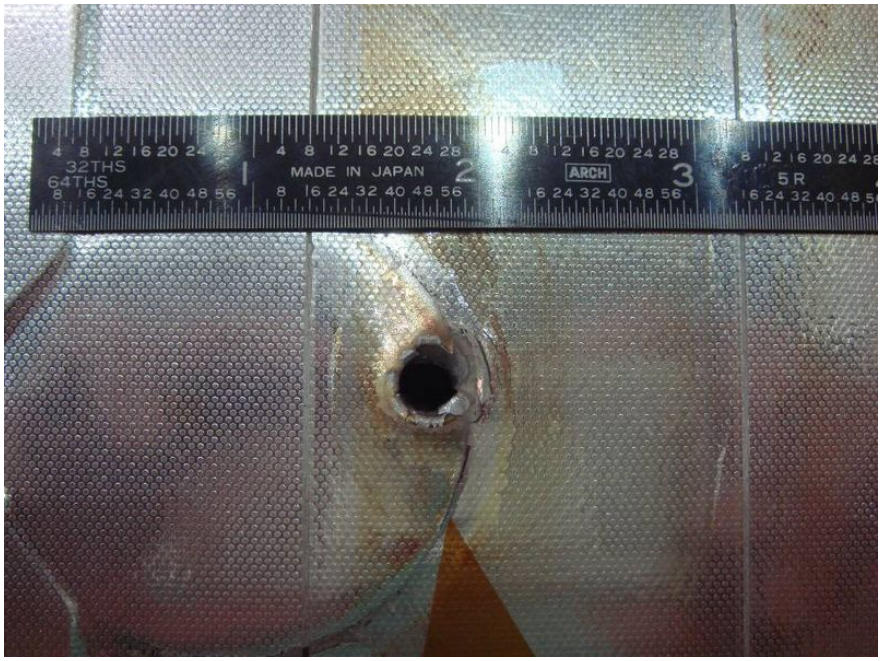


Figure 4: Debris impact on Space Shuttle Endeavorer's radiator panel[3]

action creating more and more space debris. This could result in space being inaccessible for years to come, as any object that left the earth's atmosphere would be immediately shredded by debris (thus adding even more debris).

This can be prevented by moving derelict spacecraft to graveyard orbits. However even spacecraft in graveyard orbits are not perfectly safe. Coolant tanks can puncture and coolant droplets freeze adding to the debris.

The alternative is to lower the orbit until it experiences atmospheric effects. But if the debris's orbit keeps it out of the atmosphere, then the debris must be moved.

0.3.3 CubeSats

For the task of moving debris, a CubeSat will be used. CubeSats are a type of miniaturized satellite. CubeSats get their name from their structure, they are made up of one or more $10 \times 10 \times 10$ cm units. These units have a maximum mass of 1.33 kilograms. First launched in 1998, as of August 2018, there have been 875 CubeSats launched[8].

CubeSats are used for projects that are too risky for a larger more expensive satellite,

often demonstrating new technologies. They can take on larger risks due to their low cost. As such CubeSats are common for experiential missions such as this one.

0.3.4 OSCAR

There is a CubeSat currently being designed by Rensselaer Polytechnic Institute (RPI). This CubeSat's goal is deorbit space debris by use of a magnetic tether.

This CubeSat is called O.S.C.A.R, which is short for Obsolete Satellite Capture and Recovery. It should be noted that this name is only temporary and will likely change in the near future. However for the purposes of this report, the RPI CubeSat will be referred to as OSCAR.

OSCAR will be a three unit ($10 \times 10 \times 30$) CubeSat with a mass of four kilograms. It will be able to remove four pieces of space debris.[9]

For OSCAR to be able to deorbit space debris, it must know where the space debris is.

0.3.5 Orbital Elements

This thesis is concerned with the acquisition of current orbital data of space debris and it's transformation into useful formats.

First orbital elements should be defined. Orbital elements are the parameters that define an orbit. The traditional Keplerian elements found in many textbooks are as follows:

$$\begin{array}{cc} h & \text{specific angular momentum} \\ i & \text{inclination} \\ \Omega & \text{right ascension (RA) of the ascending node} \\ e & \text{eccentricity} \\ \omega & \text{argument of perigee} \\ \theta & \text{True anomaly} \end{array} \tag{1}$$

They may be seen in figure 5.

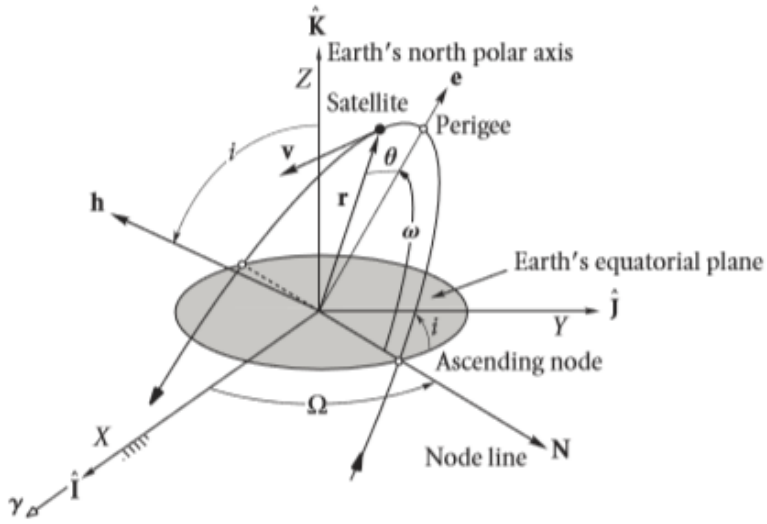


Figure 5: Geocentric equatorial frame and the orbital elements[4].

However for space debris the following set of orbital elements is slightly more useful.

e	eccentricity
a	semimajor axis
i	inclination
Ω	right ascension (RA) of the ascending node
ω	argument of perigee
ν	Mean anomaly

(2)

This is preferable as the semi major axis is used in conjunction with eccentricity with it's relation to perigee and apogee. However this is not a hard rule, and the code could be altered to use the first set of orbital elements. The orbital elements are better demonstrated in figure 6 and figure 7

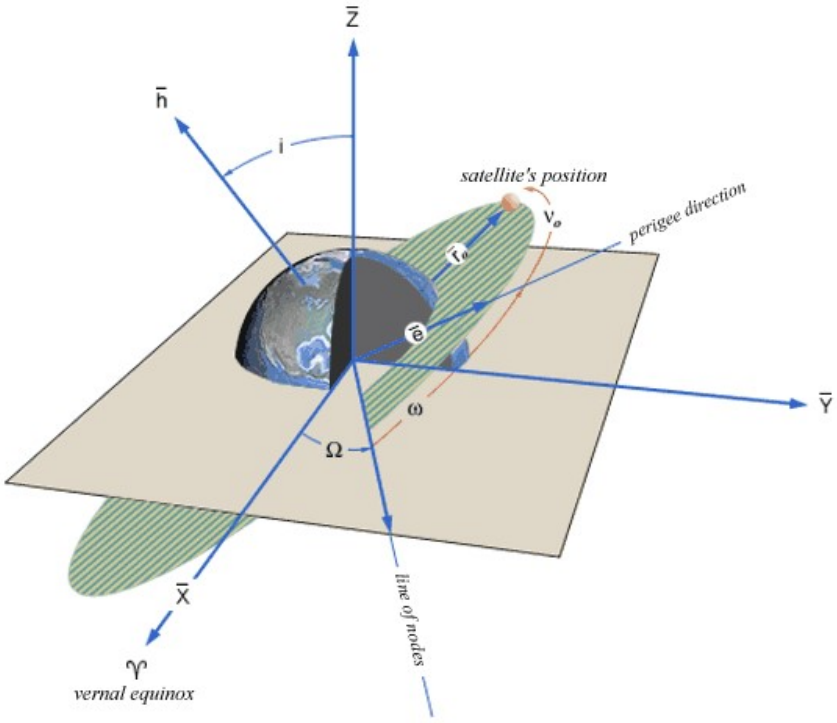


Figure 6: Diagram of Earth and Orbital Elements as well as Cartesian State Vectors[1]

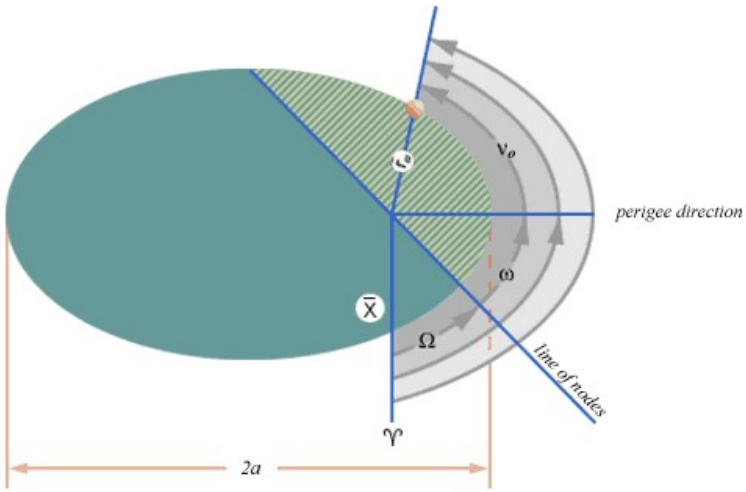


Figure 7: Diagram of Earth and Orbital Elements[1]

Table 1: Orbital Elements and their symbols[1]

Orbital Elements:		
Semi-major axis	a	Defines the size of the orbit.
Eccentricity	e	Defines the shape of the orbit.
Inclination	i	Defines the orientation of the orbit with respect to the Earth's equator.
Argument of Perigee	ω	Defines where the low point, perigee, of the orbit is with respect to the Earth's surface.
Right Ascension of the Ascending Node	Ω	Defines the location of the ascending and descending orbit locations with respect to the Earth's equatorial plane.
True/Mean Anomaly	ν	Defines where the satellite is within the orbit with respect to perigee.

0.4 Data Sources and Formats

First data sources for space debris must be found, and the data formats must be understood. The most common data format for orbital elements is the Two Line Element Set (TLE). Despite the large number of objects in space, real time data is surprisingly hard to find. While there are various websites and programs to track the location of objects such as the International Space Station, debris from old spacecraft is not quite as popular. The website space-track.org publishes TLEs for public use. First the TLE format will be explained then their acquisition will be discussed.

0.4.1 The Two Line Element Format

Orbital elements provide the means to determine a theoretical orbit. Since spacecraft are constantly experiencing forces such as atmospheric drag or solar wind their orbital elements are constantly changing.

A Two Line Element (TLE) is a data format that encodes a list of orbital elements for an Earth-orbiting object for a given point in time [Re do this]

An example shows how orbital information is derived from a Two Line Element [10].

1. **Name of Satellite:** NOAA 6

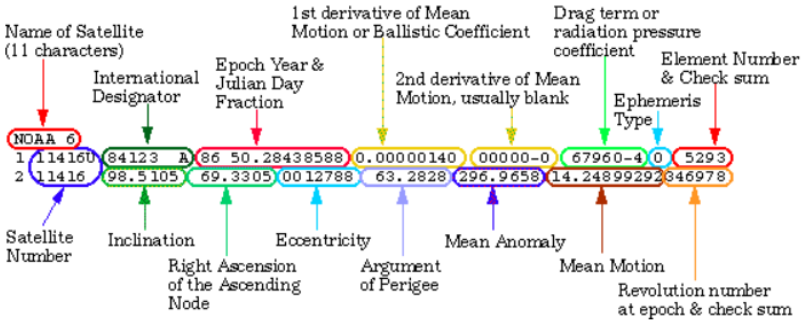


Figure 8: TLE with descriptions

This is the name of the satellite. In this example it is NOAA 6.

2. **International Designator:** 84123 A

   This number shows the last two digets of the launch year and what number launch of that year it was. In this example it is 84 123 A. The satellite was launched in the year 1984 and it was the 124th launch of the year. The A means that it was the first object that came from the launch. Check this is right

3. **Epoch Date and Julian Date Fraction:** 86 50.28438588

   The Epoch date shows the year this TLE was made and the Julian Date fraction is the number of days in said year.

4. **First Derivative of Mean Motion:** 0.00000140

   This is the daily rate of change of number of revolutions the object completes per day divided by two. This is also called the ballistic coefficient.[10] It's units are $\frac{rev}{day^2}$.

5. **Second Derivative of Mean Motion:** 00000-0

   This is the second derivative of mean motion divided by six, with units of $\frac{rev}{day^3}$.

6. **Drag term or B* Term:** 6970-4

   Of the terms given here, the B* term is the least heard of. It is a way of modeling drag on orbiting objects in propagation models. Aerodynamic drag is given by the following

equation:

$$F_D = \frac{1}{2}\rho C_d A v^2 \tag{3}$$

Where A is the area, C_d is the drag coefficient, v the velocity, and ρ is the fluid density. From Newton's second law

$$F = m \times a \tag{4}$$

the accleration due to the force of drag is

$$a_D = \frac{F_D}{m} = \frac{\rho C_d A v^2}{2m} = \frac{C_d A}{m} \times \frac{\rho v^2}{2} \tag{5}$$

The ballistic coefficient is given by the following equation:

$$B = \frac{C_d A}{m} \tag{6}$$

The starred ballistic coefficient is then

$$B^* = \frac{\rho_0 B}{2} = \frac{\rho_0 C_d A}{2m} \tag{7}$$

This turns the equation for acceleration due to drag into [11][12]

$$a_D = \frac{\rho}{\rho_0} B^* v^2 \tag{8}$$

Possibly not needed maybe take out.

7. **Element Set Number and Check Sum**

8. **Satellite Number:** 11416U

   This is the Satellite Catalog Number and designation of the object. A U means unclassified.

9. **Inclination (degrees):** 98.5105

   This is one of the orbital elements.

10. **Right Ascension of the Ascending Node (degrees):** 69.3305

    This is one of the orbital elements.

11. **Eccentricity:** 0012788

    This is one of the orbital elements.

12. **Argument of Perigee (degrees):** 63.2828

    This is one of the orbital elements.

13. **Mean Anomaly (degrees):** 296.9658

    This is one of the orbital elements.

14. **Mean Motion:** 14.24899292

    This is the orbits per day the object completes.

15. **Revolution Number and Check Sum:** 346978

A second example is given below. This time with references to the position of values, for extraction. The line under the dashes is the reference number line.

```
ISS (ZARYA)
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
------------------------------------------------------------------------
123456789012345678901234567890123456789012345678901234567890123456789
1         2         3         4         5         6         7
```

Table 2 describes the second example TLE[13].

Table 2: Description of TLE

Line 0		
Columns	**Example**	**Description**
1-24	ISS (ZARYA)	The common name for the object based on information from the Satellite Catalog.
Line 1		
Columns	**Example**	**Description**
1	1	Line Number
3-7	25544	Satellite Catalog Number
8	U	Elset Classification
10-11	98	International Designator (Last two digits of launch year)
12-14	067	International Designator (Launch number of the year)
15-17	A	International Designator (Piece of the launch)
19-32	04	Epoch Year (last two digits of year)
21-32	236.56031392	Epoch (day of the year and fractional portion of the day)
34-43	.00020137	1st Derivative of the Mean Motion with respect to Time
45-52	00000-0	2nd Derivative of the Mean Motion with respect to Time (decimal point assumed)
54-61	16538-3	B* Drag Term
63	0	Element Set Type
65-68	999	Element Number
69	3	Checksum
Line 2		
Columns	**Example**	**Description**
1	2	Line Number
3-7	25544	Satellite Catalog Number
9-16	51.6335	Orbit Inclination (degrees)
18-25	344.7760	Right Ascension of Ascending Node (degrees)
27-33	0007976	Eccentricity (decimal point assumed)
35-42	126.2523	Argument of Perigee (degrees)
44-51	325.9359	Mean Anomaly (degrees)
53-63	15.70406856	Mean Motion (revolutions/day)
64-68	32890	Revolution Number at Epoch
69	6	Checksum

to do here, add more details on what the checksum and stuff like that is

The process of TLE gathering and updating is somewhat shadowy. [14] What is known is that observations are collected multiple times per day at the Joint Space Operations

Center (JSPOC) which is operated by the US Air Force Space Command (AFSPC). Then the unclassified TLEs are passed on to space-track for public release.

0.4.2 Space-Track.org

Information about objects in space may be found the website space-track.org. Space-track is the main source for orbital data, though some are also available from the website celestrak.com. Of the two space-track has far more information and better methods of access. "Space-Track.org is managed, maintained and administered by JFSCC"[15]. Put more information about the website here.

space-track allows information to be downloaded manually from the TLE search [13] or the satellite catalog search [16]. While these are useful tools the code needs a way to automatically download data. Fortunately space-track also has an API that allows queries.

0.4.3 Space-Track Query

0.4.4 SatCat?

0.5 NORAD Space-Track

Now that the Two-Line Element data format is understood, and the the desired outcome (the orbital elements) is also understood, we need to

Describe the site desribe the SATCAT, then the TLE query

0.6 Code Overview

The code obtains orbital data for space debris depending on what variables the user has inputted, such as the launch year of the debris or how recent the data should be. First the code ascertains what data is already available. It checks to see if there is a .mat file for the

SATCAT. This is simply a file that contains the catalog ids of the debris. If this file does not exist then the code has not been run for the current parameters, and the code will be run.

The code will also check to see if there is a TLE .mat file. If this file exists the code then checks when the file was created. Depending on user variables, the code may consider the information to be out of date, will rerun everything. However if the TLE .mat file is recent, then the code will load the current debris data into the workspace.

Assuming that the code detects that it has no data, or the data is out of date the code will work in the following steps

1. The desired NORAD satellite catalog ids will be collected with the get_SATCAT.m file. These ids are determined by user variables such as launch date. The code downloads them as a csv, strips things like quotation marks away, and sorts the ids in order.
    - if a SATCAT .mat file exists and is recent this step is skipped.
2. The code then begins to loop through the array of Ids. It enters an error catch to handle url time outs.
    - Within this loop the code then downloads the TLEs from the satellite ids. The TLEs are stored in a string. It does this in groups that have their size determined by the user. A large group size may risk url timeouts, but a small group size will result in a long run time.
    - The TLEs are then parsed, the TLEs are turned into usable orbital information, and saved as an array in MATLAB.
    - When the loop has run through all the Ids, the array of data is saved as a .mat file wit the date of it's creation.
3. Now the array of debris data is ready. A copy is saved with a header listing what each collum represents. Another is sent to an orbit visualization file.

4. It should be noted that not all TLEs requested will be provided. So if the code requests TLEs of objects with a id from 1000 to 1100 and expects 100 TLEs, but some are classified, then duplicated TLEs less than 100 TLEs will be returned. The user now has a usable list of space debris and their orbital elements.

What now follows is a more in depth explanation of each file.

0.6.1 VarStore.m

This is a file where a few important variables are stored. It allows the user to only have to edit one file to change the code's operation.

0.6.2 UserPass.m

This file is where the username and password for space-track.org are stored. This is kept separate from the other code to ensure privacy of usermane and password.

0.6.3 MASTER_TLE.m

This file is the master file for the Two Line Element MATLAB files. Running it will run all of the associated MATLAB files. These MATLAB files take some time to run, so it may be convenient to alter the range of data operated on by MASTER_TLE.m in VarStore.m

0.6.4 get_SATCAT.m

Get_SATCAT.m is the MATLAB file that gets the satellite catalog numbers of all orbital debris launched after a given year and with the RCS_SIZE value equal to SMALL. The Launch Year can be set by the values given by the user in VarStore.m The default launch year is set to 1990. Note that an earlier launch year will provide more data, and thus it will take more time to process. This may cause a time out error. Should this happen the timeOutVal in VarStore.m should be adjusted to be longer.

Once the SATCAT csv file has been acquired the file then formats it. The quotation marks that are around every entry are removed. The debris that have already deorbited are also removed. Finally the debris is sorted by NORAD Catoluge Id, and saved as a .mat file.

0.6.5 get_TLE_from_ID_Manager.m

The purpose of get_TLE_from_ID_Manager.m is to act as a handler for function get_TLE_from_NorID.m.

The code first checks to see if there exists a folder to store the data in, if there is not a folder is created. Then the code begins to run through the list of NORAD Ids. It sends these Ids to get_TLE_from_NorID.m, receives the orbital element data from those Ids.

The code also contains an error handler. The most common problem with downloading large numbers of TLEs is a url timeout may occur. Timeouts are easy for a human to deal with, they may be fixed by simply refreshing the web page. However for MATLAB, it is more difficult. The code deals with a url timeout as follows:

First the code detects the specific error thrown for url timeouts. This error is caught, so the code does not stop running. Then the code gets the current location in the Id array the time out occurred at, and restarts at that spot. The effect of this is the same as refreshing a webpage that is not loading.

Once all the data has been collected for the given Ids, it is saved as a .mat file with the time that it was created.

0.6.6 get_TLE_from_NorID.m

The purpose of get_TLE_from_NorID.m is to download the TLEs from NORAD given Ids, and to save the TLEs as useful orbital elements. This function may be divided into two parts: the web interface for TLEs and the TLE to orbital element section.

The first part of the code creates a url from the current set of Ids. This url is then used to download the TLEs. The TLEs are saved as a string within MATLAB.

In the next part of the code a loop goes through the TLE string. The information from these TLEs is saved as a matrix. Earlier vesrions of this projects code first downloaded all TLEs as .txt files and then parsed through the .txt files to get the orbital elements. However that was unnecessarily slow as MATLAB had to open each .txt file and read it's contents as opposed to reading the contents of a string already in the workspace. By constantly saving the results as a .mat file through out the operation of the code, there is no threat of data loss.

From each TLE the inclination (i), RA of ascending node (Ω), eccentricity (e), argument of perigee (ω), mean anomaly, and mean motion (n) may be acquired directly. From these values the other orbital elements may be calculated.

Period of rev:

$$T = \frac{24 \times 60 \times 60}{n} \tag{9}$$

Semi-major axis:

$$a = \left(\left(\frac{T}{2\pi} \right)^2 \times \mu_{\text{earth}} \right)^{(1/3)} \tag{10}$$

where μ_{earth} is the standard gravitational parameter of earth: 3.986×10^{14}.

Semi-minor axis:

$$b = a \times \sqrt{1 - e^2} \tag{11}$$

Perigee and Apogee:

$$r_{\text{per}} = (1 - e) \times a \tag{12}$$

$$r_{\text{ap}} = (1 + e) \times a \tag{13}$$

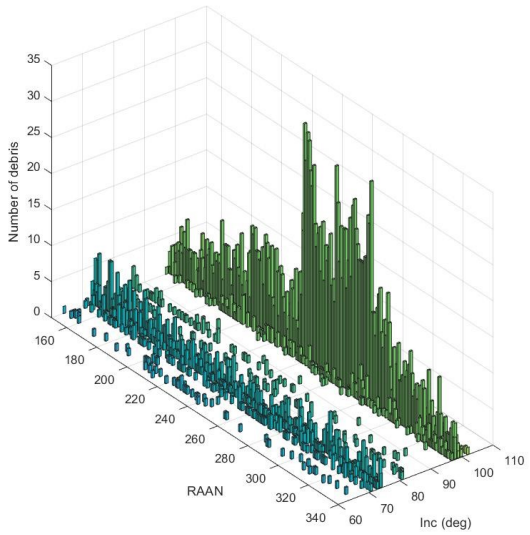


Figure 9: Debis loc

0.7 Targeting

Once the data has been collected, the next step is to target possible orbits. Since OSCAR has a delta V of Find delta V inclanation and plane changes are out of the question. What is desired is a number of debris in a plane with simialr inclanations.

0.8 Conclusion

The goal of this thesis was to create a program that can download the latest orbital elements needed to calculate the real time locations of space debris, given user inputted parameters. The information would be usable to someone who has taken a basic spaceflight mechanics class.

Bibliography

[1] NASA, 2012.

[2] NASA. Space debris impact on space shuttle window.

[3] NASA. Image of the entry hole created on space shuttle endeavour's radiator panel by the impact of unknown space debris.

[4] HOWARD D. CURTIS. *ORBITAL MECHANICS FOR ENGINEERING STUDENTS*. Elsevier Butterworth Heinemann, 2013.

[5] Swinburne University of Technology. Semi-major axis.

[6] Usnrl. Vanguard i celebrates 50 years in space, Mar 2008.

[7] How many space debris objects are currently in orbit?, Jul 2014.

[8] Erik Kulu. Nanosatellite and cubesat database.

[9] Paul McKee. Cubesat dynamics and attitude control: Kanes method, lqr, and kalman filtering. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, 2018.

[10] Amiko Kauderer Kim Dismukes. Definition of two-line element set coordinate system, 2011.

[11] By Dr. T.S. Kelso. Frequently asked questions: Two-line element set format, 1998.

[12] Canadian Satellite Tracking and Orbit Research (CASTOR). B-star drag term, 2010.

[13] Space Track. Basic description of the two line element (tle) format, 2013.

[14] David A Vallado and Paul J Cefola. Two-line element sets–practice and use. In *63rd International Astronautical Congress, Naples, Italy*, 2012.

[15] Space Track. Space track legend, 2018.

[16] Space Track. Space-track satellite catalog search, 2018.

Appendix 1 - MATLAB code

0.8.1 Master_TLE.m

```
%% Master Program
% By Philip Hoddinott
%% Setup
close all; clear all; clc; % clear workspace
%% get data
VarStore % run var store for stored variables, ugly but it works


%% check for existing data
strNam = ['mat_files/TLE_',num2str(launchYear),'.mat']; % get
    strNam
strNam_SC = ['mat_files/SATCAT_',num2str(launchYear),'.mat']; %
    get strNam

try % check the satcat tog;
    load(strNam_SC);
    tog_SC=0;

    try % try for TLE file
        load(strNam, 'tle_final','dateCreated'); % load in file
        cTime =datetime;
        if cTime>(dateCreated+calweeks(1)) % file out of date,
            rerun not SATCAT
            fprintf('File is one week out of date, auto running
                program\n');
            tog_All=1;
            tog_SC=0;
        else % file recent, no need to run any
            fprintf('The file %s, was created within the last week
                \n',strNam);
            tog_All=0;
            tog_SC=0;
        end
    catch ME % if tle not found or no date run all
        switch ME.identifier
            case 'MATLAB:load:couldNotReadFile'
                warning('TLE File does not exist, auto running
                    program');
            case 'MATLAB:UndefinedFunction'
                warning('dateCreated not found, auto running
                    program');
```

```
35          end
36          tog_All=1;
37          tog_SC=0;
38      end
39
40  catch ME
41      switch ME.identifier
42          case 'MATLAB:load:couldNotReadFile'
43              warning('SATCAT File does not exist, auto running
                    program');
44              tog_SC=1;
45              tog_All=1;
46      end
47  end
48  % assign values
49  get_SATCAT_tog =tog_SC; % toggle for get_SATCAT 1 = run, 0 = don't
        run
50  get_Multiple_TLE_from_Id_tog =tog_All;
51  readTLE_txt_tog=tog_All;
52  check_TLE_Edit_TLE_tog=tog_All;
53
54
55
56  %% func get_SATCAT
57  % Function to get a .mat file from the SATCAT
58
59  if get_SATCAT_tog==1 % if the satcat file is out of date, run this
60      get_SATCAT % get SATCAT, comment out if already run
61      fprintf('get_SATCAT.m has finished running\n'); % output
            SATCAT has run
62  else % if the satcat file is recent then no nead to run get_SATCAT
63
64      load(strNam_SC,'all_TLE','decayEnd'); % load mat of SATCAT
65      fprintf('get_SATCAT.m was not run\n'); % output SATCAT was not
            run
66  end
67
68  relDeb=str2num(char(all_TLE(2:decayEnd,2))); % get NORAD CAT ID
69
70  %% func get_Multiple_TLE_from_Id
71  % Function to get tle txt files from the norat_cat_ids in the
        SATCAT
72  VarStore % run var store for stored variables, ugly but it works
73
74  if get_Multiple_TLE_from_Id_tog==1
```

```
    get_TLE_from_ID_Manager
    fprintf('get_Multiple_TLE_from_Id.m has finished running\n');
else
    fprintf('get_Multiple_TLE_from_Id.m was not run\n');
end

%close all; clear all; % clear out everything
VarStore % run var store for stored variables, ugly but it works
strNam = ['mat_files/TLE_',num2str(launchYear),'.mat']; % get
    strNam

load(strNam, 'tle_final')

%tle_low=sortrows(tle_final(:,:),11);
%save('Orbits_MOD_1/tle_low2high.mat','tle_low');
%{
tle_high=sortrows(tle_final(:,:),11,'descend');
save('Orbits_MOD_1/tle_high2low.mat','tle_high');
% Note that these files could be made functions in MATLAB. For
    debuggin
% purposes they currently are not

tle_high=sortrows(tle_final(:,:),4,'descend');
save('Orbits_MOD_1/tle_RANN.mat','tle_high');

tle_INC=sortrows(tle_final(:,:),3,'descend');
save('Orbits_MOD_1/tle_INC.mat','tle_INC');
%}
tle_view=tle_final;
tle_view_temp=["norad_cat_id","Epoch time","Inclination (deg)","
    RAAN (deg)","Eccentricity (deg)","Arg of perigee(deg)","Mean
    anomaly (deg)","Mean motion (rev/day)","Period of rev (s/rev)
    ","Semi-major axis (meter)","Semi-minor axis (meter)"];

tle_veiw = [tle_view_temp;tle_view]; % useful for looking at
    numbers
```

0.8.2 get_SATCAT.m

```
%% Gets SATCAT from NORARD Query
% based off this code:
% https://github.com/jgte/matlab-sgp4/blob/master/get_tle.m

load('UserPass.mat') % load in username and password

```

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

post={... % Create Post (rember to referance the github)
  'identity',username,...
  'password',password,...
  'query',[...
    'https://www.space-track.org/basicspacedata/query/class/satcat
      /OBJECT_TYPE/debris/',...
    'RCS_SIZE/small/LAUNCH_YEAR/>',num2str(launchYear),'/'...
    'orderby/DECAY asc/format/csv/metadata/false',...
  ]...
};


out=urlread(URL,'Post',post,'Timeout',timeOutVal); % gets the
   output
outStr=convertCharsToStrings(out); % coverts output to string


newStr = strsplit(outStr,["\n"]); % split string by line break
for i=1:length(newStr) % split string by commas
    all_TLE(i,:) = strsplit(newStr(i),',');
end


[m,n] = size(all_TLE); % get size of matrix
%% Remove " marks
for i=1:m % remove the " marks
    for j=1:n
        all_TLE(i,j)=strip(all_TLE(i,j),'left','"');
        all_TLE(i,j)=strip(all_TLE(i,j),'right','"');
    end
end

%%  find where the last decayed item is and remove it
decay_loc=8;

for i=1:length(all_TLE(1,:)) % find the columm decay is located in
   , it should be in 8
    if all_TLE(1,i)=="DECAY"
        decay_loc = i;  % save decay loc
    end
end


```

```
49 decayEnd=m+10;
50 for i=2:length(newStr) % find the columm decay is located in, it
       should be in 8
51     if all_TLE(i,decay_loc)~=[""] && decayEnd==m+10
52         decayEnd=i-1; % get the last location where there is no
               yet decay
53     end
54 end
55 all_TLE=all_TLE(1:decayEnd,:); % trim to only have debris that has
       not yet decayed
56
57
58 all_TLES=sortrows(all_TLE(2:end,:),2); % sort rows by NORAD ID
59 all_TLE=[all_TLE(1,:);all_TLES]; % save the sorted row by NORAD ID
60
61 %% save to a file
62 strNam = ['mat_files/SATCAT_',num2str(launchYear),'.mat'];
63 save(strNam,'all_TLE','decayEnd');
```