Python Based Plume Dynamics Estimation Tool (PyPDET)

Rapid Plume Strike Analysis for RPOD Maneuvers in Deep Space Operations

Student Name: Andres Torres-Figueroa
Academic Level: Undergraduate (Senior)
Academic Major: B.S. Aerospace Engineering
Academic Institution: University of Central Florida

Mentor Name: Dr. Jonathan Pitt
Mentor Job Title: Subject Matter Expert, Computational Fluid Dynamics
Org Code/Branch: *Deep Space Logistics (DS-000) (Aegis Aerospace Inc.)*
Division: *Advanced Exploration Systems (AES)*
Directorate: *Human Exploration and Operations (HEO)*

1.0 Abstract

I worked as a NASA Intern during the Summer 2023 term in the DS-00 division under the supervision of my mentor, Dr. Jonathan Pitt. Our goal was to build on our previous work from 2022 to develop a plume strike estimation tool using a prescribed physics methodology and model plume impingement effects while considering the dynamics of a rendezvous, operations, proximity, and docking (RPOD) maneuver. This tool supports previously configured CFD-DSMC calculations by allowing for rapid analysis of initial designs using a low-fidelity source flow model. Engineers can then use the high-fidelity CFD-DSMC tool to consolidate results as they work towards finalizing a design.

This year's project was focused on developing a software application that other engineers would be using in their analysis. Thus, the user's experience was considered in the development of this application. Proper documentation, testability, and modularity of the codebase was our priority. For example, the project included auto documentation procedures to start building towards a User Manual, while also including dedicated demonstration cases for more explicit communication of functionality. Also, this project included a framework for testing the source code for future developments. Additionally, care was taken to develop the code using an Object-Oriented Programming approach. Thus, allowing for a modular extensibility of functionality in anticipation of future developments.

The core work of this project was developing an algorithm that would transform the visiting vehicle and associated thruster data according to the kinematics described in the jet firing history. It would then calculate the estimated plume strikes on two of the target vehicles and write data accordingly into a VTK file. Summer work is to conclude by developing and presenting a PowerPoint slide deck at the intern exit briefing on August 11th, 2023. Once the model for simple plume strike calculations is developed and tested there are several avenues to explore to continue development of this tool. These are also discussed in this report.

2.0 Methodology

Quick descriptions for the relevant technical choices made. Mainly two categories are covered. First is the modules used within Python to meet the desired requirements. Secondly, the logic of the plume strike model is also discussed at a high level.

2.1 Python Module Selection

Requirements	**Design Decision**
Program needs to be modular in complexity on of analysis, run on an HPC environment, user, and developer friendly, and be able to interface various data sources.	Python chosen as programming language.
Program needs to have embedded documentation for developer support	Pdoc is a Python module

Program needs to read in and parse text files containing Jet Firings Histories (JFH) and Thruster Configuration Data (TCD) relevant to the RPOD scenario.	Python can handle using standard libraries for string manipulation.
Program needs to read in surface mesh data for a target and visiting vehicle to define an RPOD scenario, while also transforming the STL data according to the TCD and JFH data.	numpy-stl is a library within Python that is used.
Program needs to write resulting data to files in a VTK format for easy visualization of results.	Export VTK is another library within Python is used (pyevtk)
Program needs to allow user to configure plume physics, thruster operating conditions, mesh data, and meta data for and RPOD scenario.	Python has configuration objects, and uses Object Oriented Programming

2.2 Plume Strike Models

This initial development of this tool call for the simplest representation of plume model that is equivalent to a "line of sight analysis". It does not calculate physics-based plume dynamics but does execute calculations that sets up a framework for easy implementation of plume dynamics equations. Mainly, it loops through the relevant STL surface elements and calculate needed parameters for equations used to such as pressure, mass and contamination flux, and heat transfer.

Three logical conditions were analyzed to determine whether a face is considered struck or not. They are listed below.

1. Is the distance to the face within the user defined "critical distance" of the active thruster?
2. Is this distance vector we calculated within the user defined "half angle" value of the plume cone, i.e., how big is the angle between the distance vector and the plume's centerline vector?
3. Is the current STL element facing the same direction of the plume? This is determined by evaluating the dot product between the STL elements normal vector, and the plume center line vector. Note that the target vehicle STL file must have outward facing normal vectors.

3.0 Demonstration Cases

This section outlines simple demonstration cases used to communicate what a proper calculation should look like. One simplification to consider is that for these runs, one thruster is turned on at each time step. We can interpret cumulative strikes as each time step for which a thruster struck the face. The idea is to use this configuration as a baseline for visual confirmation of a proper calculation.

3.1 Flat Plate Fly By

This case serves a subscale test of the plume strike algorithm. The STL surface for the flat plate contains 2000 elements and 50 timesteps were simulated. This is an extremely lightweight case and runs in only 9 seconds. From the results we can see all three conditions of the plume strike model working as intended. In particular, we see the face orientation check working as intended. Since faces not facing the active thruster do not have any strikes record at all. Figure 2 shows the faces along the edge of the plate are not getting hit, and Figure 3 shows that faces at the bottom of the plate are also not getting hit.

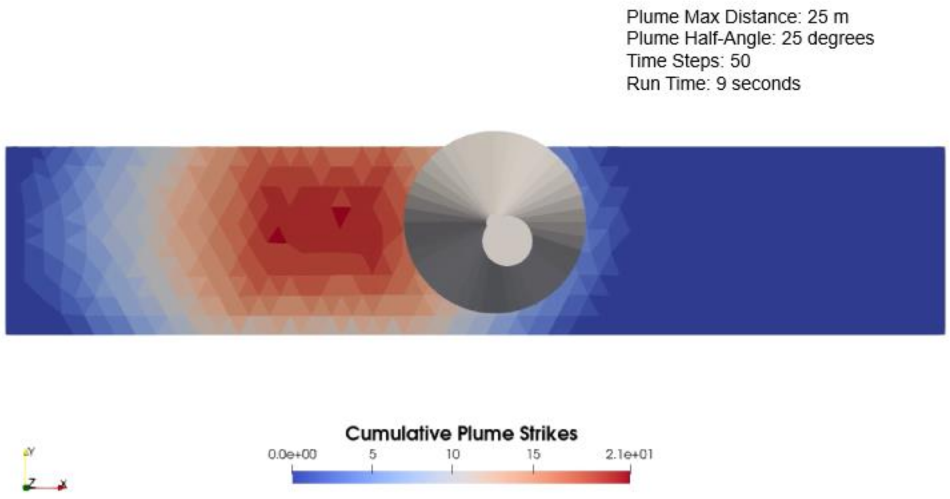

Figure 1: Z-View

8

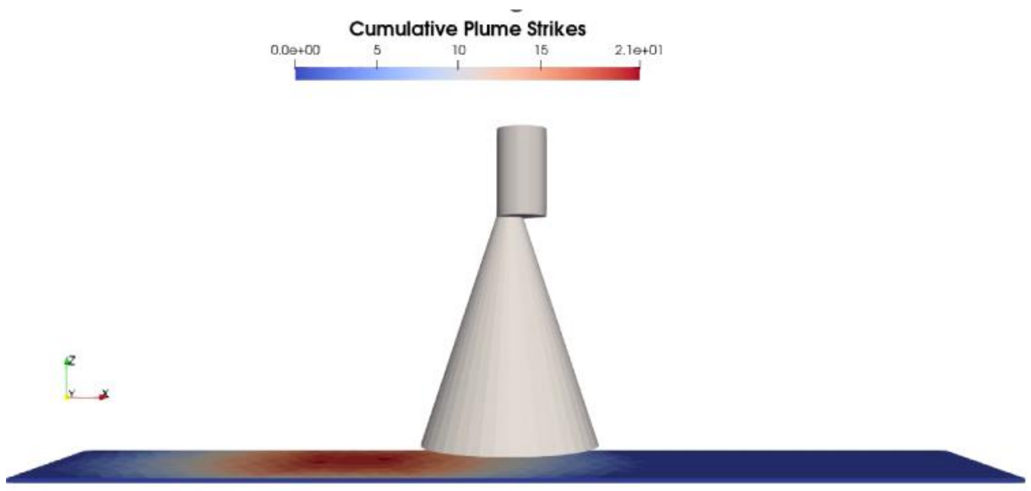

Figure 2: Y-View

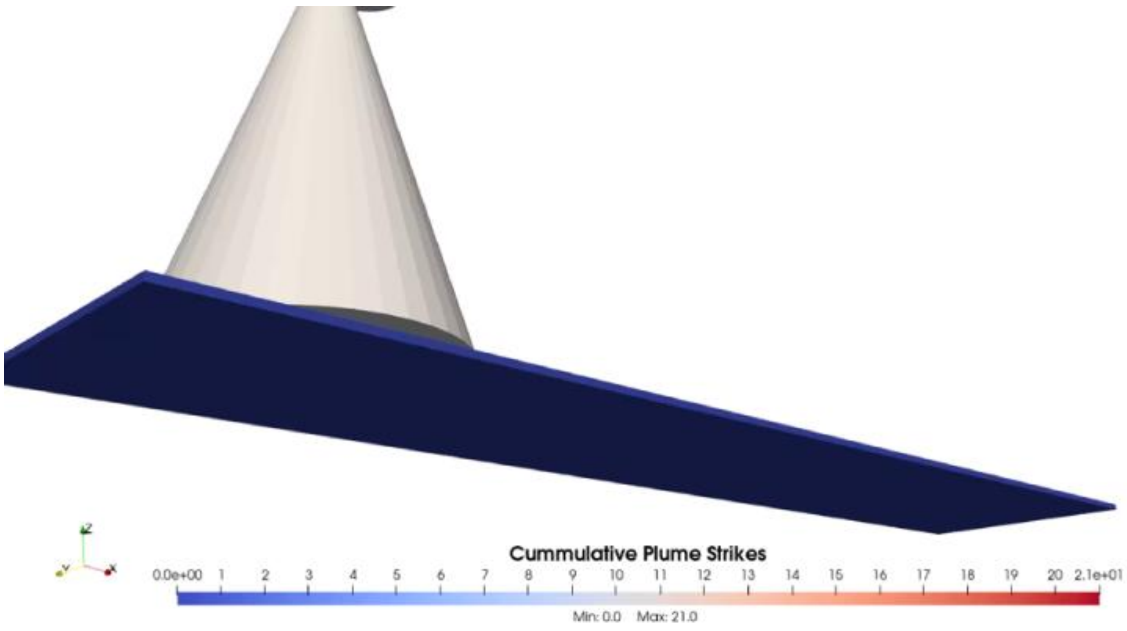
Cummulative Plume Strikes
0.0e+00 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 2.1e+01
Min: 0.0 Max: 21.0

Figure 3: Bottom-View

3.2 ISS Fly Bly Case

This case introduces the ISS STL mesh (75,000 elements) and increases time step count to 100 steps. This simple addition of complexity give a better idea of the runtime that can be expected when running full RPOD analysis. This case took around 5 minutes to run on a single core. This is still extremely fast, as a similar case could require several hundred core-hours to run using DSMC solvers.

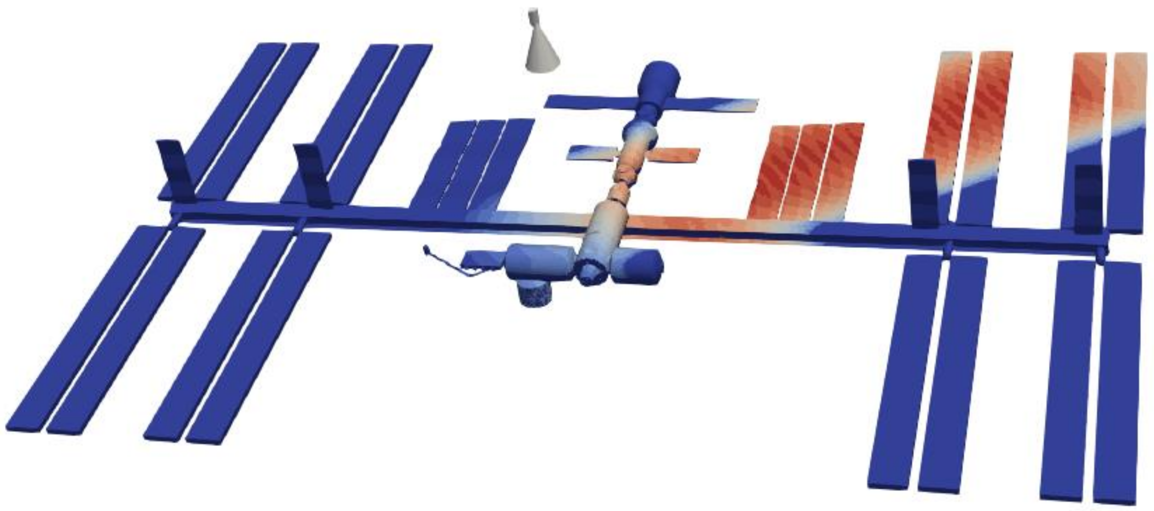

Figure 4: ISS Fly By Midway Shot

3.3 ISS Notional Docking

This section introduces two cases that are used to demonstrate how this tool can help characterize impinging plume dynamics and quickly differentiate these effects according to the supplied jet firing history. Two cases are selected to compare trajectories when docking to a port. Figure 5 provides a diagram for the first case, which approaches the port in a horizontal line representing a "head-on" trajectory. Similarly, figure 7 shows a diagrams for the second case, which approaches the docking port approaches in a descending straight-line trajectory representing a "overhead-descending" approach.

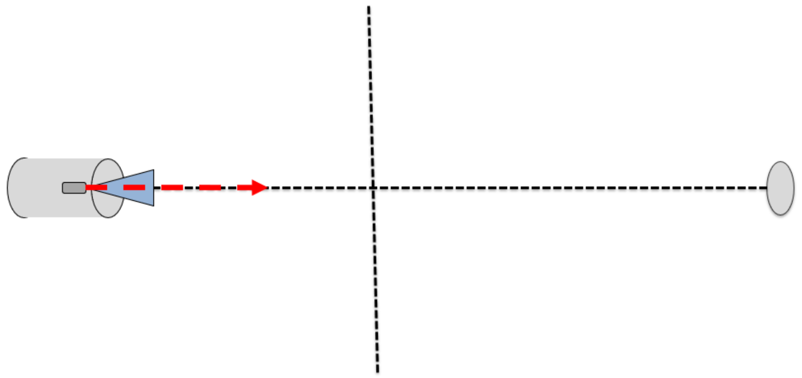

Figure 5: Diagram Showing "Head-On" Docking Trajectory

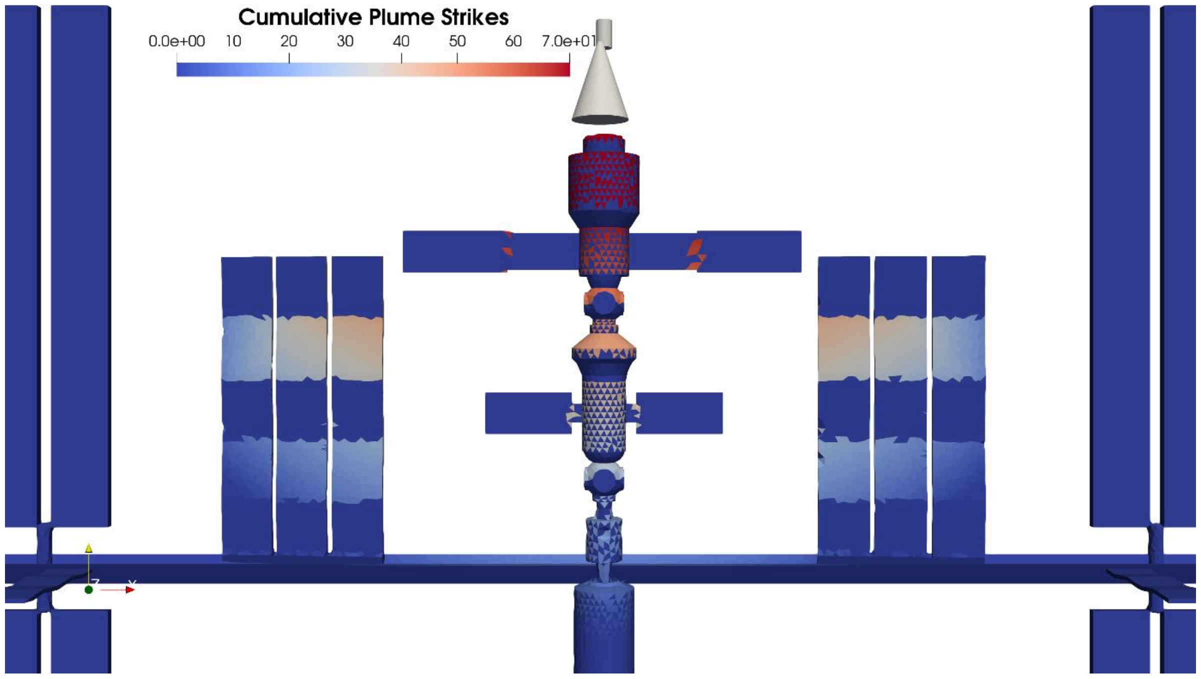

Figure 6: Resulting Strikes From "Head-On" Docking Trajectory

From results seen in Figure 6 it is easy to identify which regions of the target vehicle's surface are affected the most. Since this trajectory is straight on the port junction itself is getting blasted, experiencing 70 total strikes. However, we can see that this is only experienced on surfaces that are facing the active thruster. Many of the solar panels have orthogonal faces and are not getting struck. This is especially apparent with the solar panel arrays that are connected to the main truss. Their "waffled" configuration means that man of the faces are facing away from the active thruster, and are thus, not getting struck.

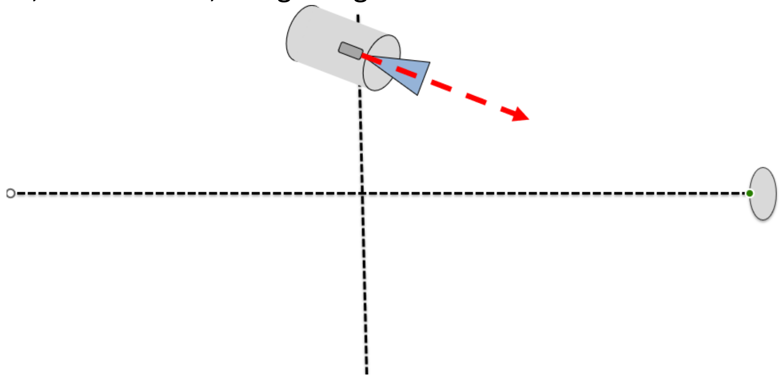

Figure 7: Diagram Showing "Overhead-Descending" Docking Trajectory

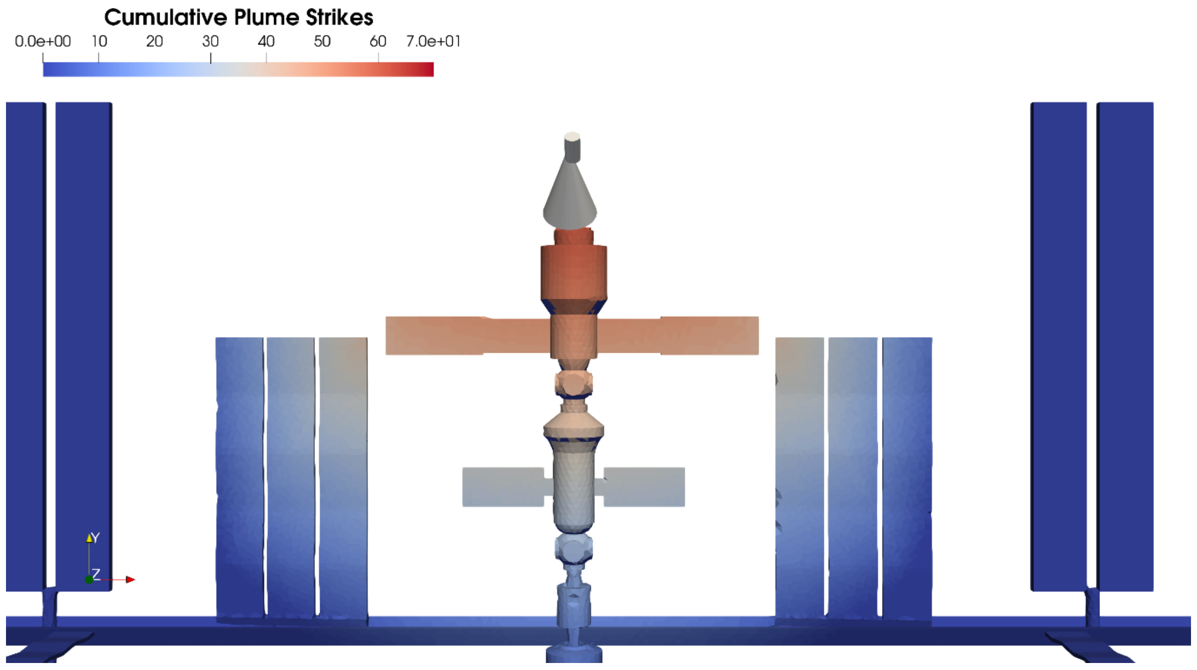

Figure 8: Resulting Strikes From "Overhead-Descending" Docking Trajectory

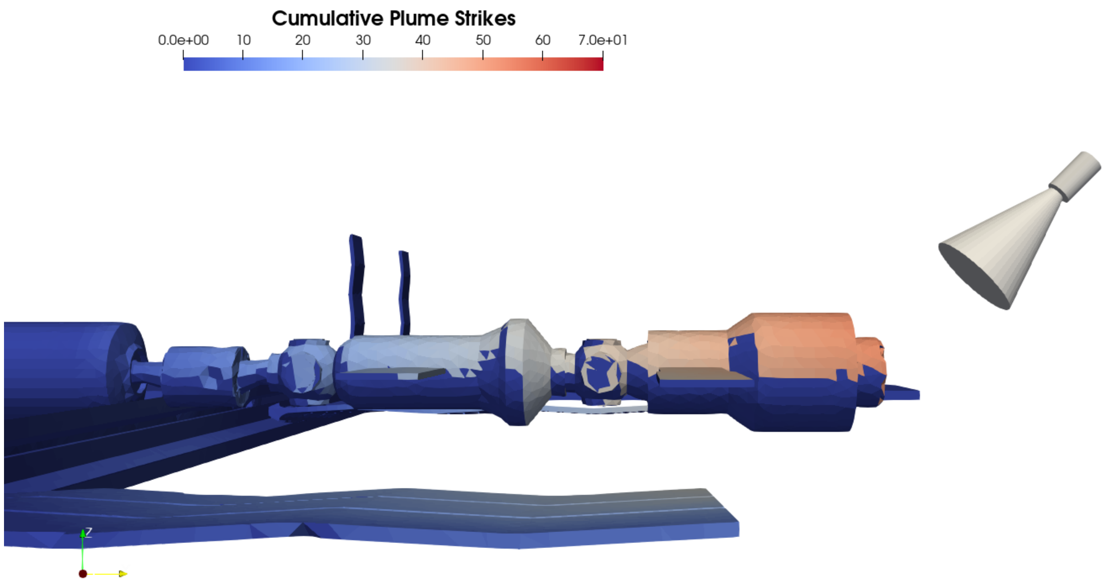

Figure 9: Side Profile of "Overhead-Descending Results

As mentioned, Figure 7 shows a similar trajectory and docking to the port. However, this "overhead-descending" produces distinct results to the "head-on" approach. Figure 8 displays these results. From here we can see that the port to dock to is still getting the most strikes at 70 total strikes. However, we see a stark difference in which faces are being hit. The overhead approach means that a different set of faces are being hit. In this case it is most of the faces on the upper surface. We can see these difference in strikes on the surrounding solar arrays; especially the "waffled" array. All the faces are facing the active thruster and thus are being hit. Figure 9 helps further illustrate this point. It can be seen that faces on the top half of the ISS surface are being struck while faces on the belly are not getting struck at all.

4.0 Discussion and Next Steps

These results represent a minimal viable product and are encouraging for continuing to build this tool. Mainly the runtime and logical output of calculation indicate that the envisioned tool is feasible. Further development should be aimed towards adding resiliency to the current code base and adding features for better analysis. The table below briefly describes proposed next step for further development of PyPDET.

1. Further Development of Test Harness
    a. Fill in testing gaps of source code.
    b. Develop output comparison scripts
    c. Re-factor simulation logging framework using built in python logger.
2. Add Appropriate Physics Models
    a. Equations for Mass, Contamination, and Heat Flux

   b. Equations for Pressure and Shear Loads
   c. Target Vehicle Dynamics from Plume Pressure Loads
   d. Shading Algorithms
3. Data Processing Routines
   a. Normalize Physics Calculations
   b. Conduct Strike Counts According to Component ID
   c. Determine Thruster Responsible for the Most Damage
4. Engineering Design Routines
   a. Trade studies using Jet Firing History and Thruster Configuration data.
   b. Two-way coupling of Target Vehicle and Visiting Vehicle plume dynamics.

5.0 Acknowledgments

I would like to thank my supervisor Dr. Jonathan Pitt. I can't put into words how valuable your guidance and patience has been over the last couple of years. Through your influence and excellent example, I have made significant progress towards becoming the problem solver and professional I want to be. To Dr. Michael Kinzel, your continued commitment to the success of your students does not go unnoticed. For me personally, I could never thank you enough for all the mentoring and support over the years. I would also like to thank Mr. Shirish Patel, for advocating for me and the other interns on center. Your eagerness to get to know, connect, and help those around you is something I won't forget and is an example I hope to follow.