OLIN UNDERGRADUATE RESEARCH JOURNAL

Volume 1
2023

Olin College of Engineering

Olin Undergraduate Research Journal

April 2023

Thank you to the contributing authors and reviewers who helped make this journal happen:
Contributing Authors: Charlie Babe, Sreenidhi Chalimadugu, Avery Clowes, Lily Dao, Ben Eisenbraun, Kaitlyn Fleming, Kristtiya Guerra, Andrew Haviland, James Jagielski, Anargyros Kriezis, Christopher Lee, Audrey Lee, Celvi Lisy, Whitney Q. Lohmeyer, Bryce Mann, Kate McCurley, Braden Oh, Andrew Phillips, Michael Remley, Michael S. Rugh, and Carlota de Huelbes Ramiro
Reviewers: Kaitlyn Fleming, Abby Omer, Neel Dhulipala, Whitney Q. Lohmeyer, and Jess Charlap

Contents

Part I

A Low-Cost, Active Tracking Approach for an Antenna Control Unit (ACU) Ground Station

A Low-Cost, Active Tracking Approach for an Antenna Control Unit (ACU) Ground Station

Carlota Ramiro de Huelbes* and Phillip Post †
Olin College of Engineering, Needham, MA, 02492

Celvi Lisy‡
Olin College of Engineering, Needham, MA, 02492

Anargyros Kriezis§
Stanford University, Stanford, CA, 94305

Sreenidhi Chalimadugu¶
Olin College of Engineering, Needham, MA, 02492

Bryce Mann‖
Olin College of Engineering, Needham, MA, 02492

Whitney Q. Lohmeyer**
Olin College of Engineering, Needham, MA, 02492

An unprecedented number of communication satellite constellations will be deployed in the next decade, as companies like Amazon and SpaceX aim to provide global Internet connectivity. One of the fundamental elements of the financial success of such constellations is the availability of a low-cost user terminal (UT) solution. This paper presents the design of a low-cost UT antenna control unit (ACU), developed using commercial off-the-shelf (COTS) components, capable of tracking satellites in Low Earth Orbit (LEO), Medium Earth Orbit (MEO), and Geostationary Orbit (GEO) via multiple modes of operation with remote control access via a user interface. The ACU employs a closed loop, active tracking python algorithm based on detected signal strength to calculate the tracking path for the antenna. Two scheduler functions were used to first acquire the satellite before handing off to this algorithm. The combination of programs served as a minimum viable proof of the affordable electrical and mechanical systems. The mechanical steering of the antenna is handled by two bipolar stepper motors with bevel gears with one motor adjusting ACU yaw, and the second motor adjusting pitch via a custom-made u-frame bracket.

I. Introduction

In an effort to provide global connectivity, several companies aim to deploy communication satellite constellations. Satellite systems like SpaceX's Starlink, Amazon's Kuiper, OneWeb, and others are developing mega constellations with hundreds of satellites orbiting in Low Earth Orbit (LEO) providing broadband Internet [1]. The cost of satellite internet from non-geostationary networks is still high, primarily dictated by user terminals cost. User terminals are the terrestrial antennas located at the user's premise, such as a home, vessel, or aircraft. The challenge with antennas for non-geostationary satellites is that they need to have a mechanism to track satellites passing overhead.

This paper outlines the design and testing of a user terminal antenna control unit (ACU) developed to provide a low-cost solution that is capable of scanning and tracking non-geostationary satellites, utilizing a driving pan-tilt

*Electrical and Computer Engineering Student, Olin College of Engineering, 1000 Olin Way Needham, MA
†Electrical and Computer Engineering Student, Olin College of Engineering, 1000 Olin Way Needham, MA
‡Mechanical Engineering Student, Olin College of Engineering, 1000 Olin Way Needham, MA
§Aerospace Engineering PhD Candidate, Stanford University, Stanford, CA
¶Electrical and Computer Engineering Student, Olin College of Engineering, 1000 Olin Way Needham, MA
‖Electrical Engineer, Olin College of Engienering, 1000 Olin Way Needham, MA
**Assistant Professor of Engineering, Olin College of Engienering, 1000 Olin Way Needham, MA

mechanism, and commercial-off-the-shelf (COTS) parts. The ACU system consists of three subsystems: tracking software, a mechanical system, and an electrical system. The motor drive consists of a pan-tilt mechanical fixture and two driving motors controlled by a Raspberry Pi. The tracking system employs a hybrid tracking algorithm with signal feedback from the antenna to detect and track the satellites by driving the motors to the optimal orientation.

II. Tracking Methods

The purpose of a satellite tracking system is to ensure a connection between an antenna control unit on the ground and a satellite in space is maintained without service interruption [2]. The main challenges with implementing a successful tracking algorithm include overcoming environmental interference, achieving high tracking accuracy, and minimizing mechanical wear [2]. All three of these barriers must be resolved or minimized either through software or hardware level changes to ensure high-performance communication services. Environmental interference includes inclement weather, wind, and other external physical disturbances [3]. ACUs are often required to maintain pointing accuracy to within a fraction of a degree for stable communication, and any external disturbance could cause critical degradation of the received signal or loss of the received signal altogether [4] [2]. This high accuracy requirement is a large hurdle as a tracking system has to maneuver an antenna along a smooth trajectory, predicting the future movement of the orbiting object, and adjusting for factors such as orbital drift [3]. Lastly, mechanical wear is a consideration for any tracking technique that relies on moving parts as excessive wear can lead to premature failure, higher costs, and increased maintenance [3]. Although there are a plethora of tracking techniques, this paper explores ones that are most commonly employed of both open and closed loop nature, including two-line element (TLE) tracking, step tracking, electronic beam steering, and monopulse tracking. The tracking algorithm implemented for the ACU was intended to function as a proof of concept of the low-cost hardware, but, as the choice of tracking technique fundamentally alters the design and cost of an ACU, it was paramount to explore all options.

A. Open and Closed Loop

ACUs can implement one of two tracking approaches: closed and open loop. A closed loop system is one in which the ACU identifies a signal, feeds the signal strength and direction of signal strength change to the control unit, and then the control unit shifts the antenna towards the higher perceived signal strength, forming a feedback loop as the process is repeated [2]. This method has the advantage of not requiring satellite position data and has a relatively simple implementation. However, this method does have the general disadvantages of shadowing, overcorrecting, and blocking effects [2]. Overcorrecting is when the control unit calculates a higher velocity than needed, leading to overshoot and potential signal degradation. Shadowing, also referred to as ghosting, is when the ACU lags behind the signal, leading to potential signal degradation. Blocking occurs when the satellite signal becomes lost due to an obstruction between the ACU and satellite [5]. All three of these issues can lead to the ACU losing the satellite and entering search mode if not properly mitigated [2].

An open loop differs from a closed loop as it relies on satellite position coordinates and position sensors such as a gyroscope to follow a predetermined path [2]. Open loop benefits from continuous tracking regardless of external factors, but this approach can also be a disadvantage as an open loop system will follow the input position of a satellite even if these positions are erroneous and produce no signal [2]. For the first "alpha" prototype of the ACU, a pure open loop tracking algorithm was developed by integrating a magnetic position sensor for tracking inputted Two-line elements. Yet, the magnetic position sensor was later abandoned as the electromagnetic fields produced by the bipolar stepper motors caused interference, rendering the sensor useless. Although not realized in the alpha prototype, well-implemented open and hybrid loop systems, ones that integrate both closed and open tracking techniques, provide tracking benefits and opportunities that will be pursued in future research.

B. TLE Tracking

Two-line element (TLE) is a satellite position and identification system developed and updated by the North American Aerospace Defense Command (NORAD) [6]. Within two strings of numbers and letters, the Keplerian elements, satellite identification, and other important details such as radiation pressure coefficient are contained [7]. Two-line element data can be input into an ACU where they are then processed, compared to the ACU's Global Positioning System (GPS) coordinates, and then utilized for the basis of ACU tracking. TLE elements serve as the backbone for open loop tracking systems.

C. Step Tracking

Step tracking, also called extreme tracking, is a prevalent method in establishing Earth to satellite communication [8]. As summarized by Figure 1, step tracking consists of four major phases: first axis shifting, peak identification, second axis shifting, and final peak identification to find the exact position of the incoming signal [3].

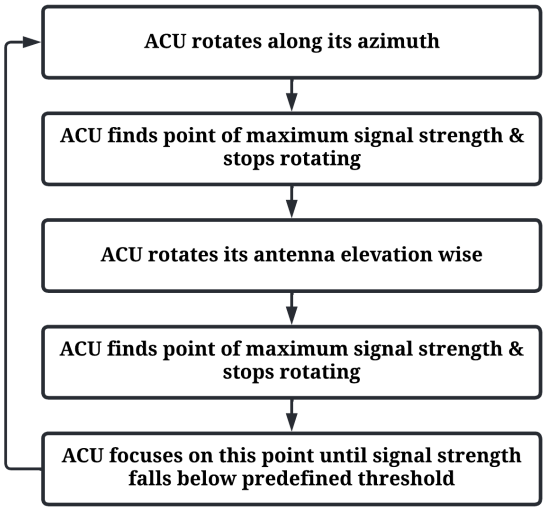

Fig. 1 Main Stages of the Step Tracking Process.

In the first step, the ACU moves along the azimuth axis until further motion in either direction causes the signal to weaken; this is considered the peak signal of that axis. Once this peak is identified, the ACU then starts shifting along the elevation axis until another peak is identified. Now that both peaks are identified, the ACU focuses on that location until the signal weakens beyond a certain preset threshold, causing the ACU to start step tracking again. This process is illustrated in Figure 2.

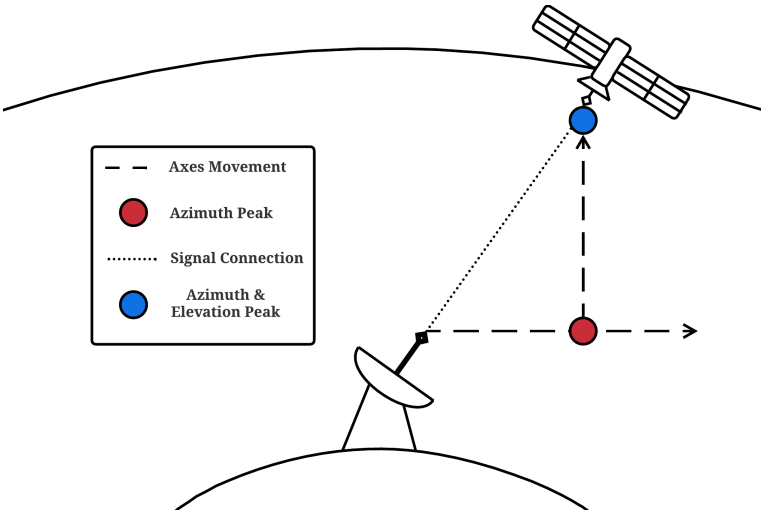

Fig. 2 Step Tracking Finding Signal Peak Along Azimuth and Elevation Axes.

The advantages of using step tracking are as follows: "simple design, easy realization, low cost, and low requirement of hardware structure" [8]. Since most step tracking algorithms use constant step size and acquisition speed, it is difficult to tune a step tracking algorithm to be completely accurate or precise, especially when affected by external factors [8]. The fundamental trade-off of step tracking is that to have a faster acquisition speed a lower tracking accuracy must be

expected and vice versa [8]. Step tracking also has the disadvantage of resulting in high wear on both the motors and gears as the constant step size requires near-continuous motion [3]. Step tracking algorithms offer tracking accuracies of 0.01^o, which are on par with fine-tuned, non-active tracking steering methods as well as conical scanning, but fall short of electronic beam steering and monopulse tracking, which can achieve 0.005^o of accuracy [3]. Overall, step tracking is an accessible tracking method that only requires the ability to move along two axes and measure signal strength, but falls behind other methods in terms of accuracy and mechanical wear.

D. Electronic Beam Steering

Electronic beam steering is an iteration upon beam forming where multiple antenna element's signals are combined, so the sum of each antenna component forms the desired beam. Beam steering uses this summative principle to change the phase and power of each element to steer the combined beam in a certain direction while attenuating any unwanted emissions in other directions. This tracking method most commonly, but not necessarily, uses a phased array antenna as they offer high accuracy and a low profile nature [9]. The main appeal of electronic beam steering is that it does not require any mechanical components as the shifting of the beam is done completely electronically, allowing for cost-saving as theoretically no mechanical maintenance is required [10]. This method is also not constrained by any mechanical limitations on the signal acquisition and tracking speed, meaning the only limiting factor is signal processing and computing time [10]. Electronic beam steering's accuracy approaches that of monopulse with an accuracy of around 0.005^o, which is more accurate than the typical 0.01^o of step tracking [3]. One major weakness of this tracking technique is that its omnidirectional nature requires the gain to be increased to produce a narrow beam with high enough signal strength and transmission rates to communicate with a satellite [3]. A higher gain means the connection becomes more sensitive to any changes in pointing error, especially external factors such as shaking or wind [4]. In addition, an electronic beam steering system requires line of sight for adequate signal communication with any blockage potentially ending normal operation [11]. Another disadvantage of this technique is the significantly higher cost, mainly due to electronic complexity [11]. Therefore, electronic beam steering provides exceptional tracking performance with no mechanical limitations but suffers from a high upfront cost and susceptibility to interference from external factors. As the main goal of this ACU is to lower cost, the economical limitations of this method make it unideal for this specific application.

E. Monopulse Tracking

Monopulse tracking, as its name suggests, calculates a satellite's position from a single pulse with slightly displaced receivers [12]. Monopulse tracking systems can have a myriad of receiver arrays with four-horn monopulse being one of the most accurate among them [12]. After a signal is received, the difference in power between the opposite receivers is used to extrapolate the real-time location of the satellite with a difference of zero implying that the ACU is perfectly aligned with the satellite [12]. The main advantage of monopulse tracking is its exceptional reliability and accuracy, typically 0.005^o, which is the best among active tracking techniques [3]. Monopulse tracking also exhibits a fast dynamic response time as it is only limited by the time needed to process the satellite position and shift the antenna into position [3]. The main drawback to monopulse tracking is its significantly higher costs compared to other tracking techniques, as monopulse tracking requires multiple receivers [3]. This higher cost is coupled with the fact that more receivers lead to a higher weight, which demands stronger motors and imposes higher inertial forces on the ACU.

F. Alpha Prototype ACU Tracking Method

The minimum viable tracking method developed for the alpha ACU system uses primarily a closed-loop, modified active step tracking technique coupled with two programmed steering functions: signal sweep and scheduler. A closed loop approach was chosen for simplicity of implementation and flexible setup. Active tracking of the satellite was handled by the unique step tracking algorithm detailed in the next paragraph whereas the two pre-programmed steering functions were implemented to improve acquisition time and accuracy. The signal strength function takes in a set of IQ signal readings from the Pluto Software Defined Radio (SDR) at a specific center frequency wavelength, set by the Raspberry Pi upon startup. The function takes a Fast Fourier Transform (FFT) of the signal readings to convert the signal into the frequency domain, which is then repeated 250 times, and averages the result to significantly reduce noise. These 250 samples are collected via a loop in the algorithm, meaning the period of data collection is variable for each trial. Finally, the function calculates the signal-to-noise ratio (SNR) as the strength of the highest peak (in dB) minus the average noise value (in dB). The noise strength is estimated by taking the average value of the frequency spectrum in

the entire sample (in dB). A notable drawback of using this method is that it cannot distinguish between two satellites with the same beacon frequency. For future work, the best method of correcting this will likely be to read the content of the beacon signal and use that to determine which satellite is currently being tracked. An illustration of the parts that facilitate this process can be seen in Figure 3.

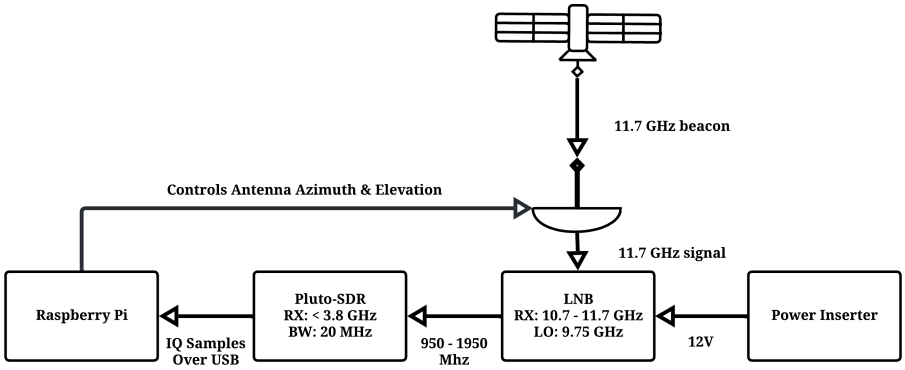

Fig. 3 Radio Signal Pre-Processing

The modified step tracking algorithm used in the alpha prototype served as an abridged traditional step tracking algorithm. The primary steps of the modified tracking algorithm are listed in Figure 4.

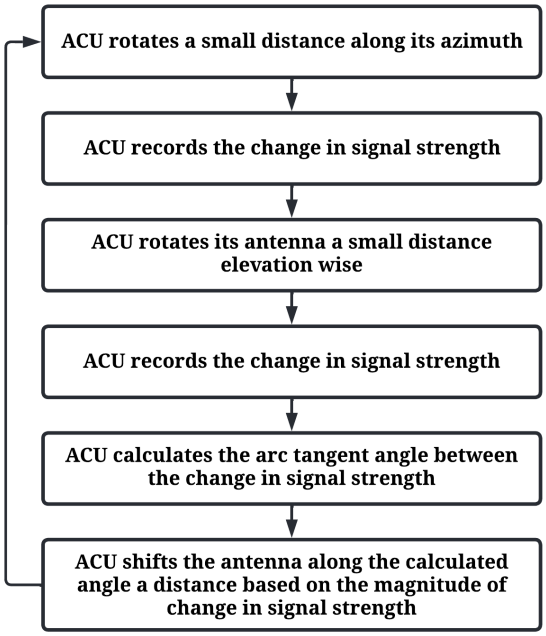

Fig. 4 Modified Step Tracking Process Steps

Whereas a traditional step tracking algorithm goes along both axes until the maximum signal strength is found, the modified algorithm travels a short distance along the azimuth and elevation axes (pan and tilt) before using the measured difference in signal strength to extrapolate a direction to track. A diagram of this process is shown in Figure 5 with accompanying Equation 1.

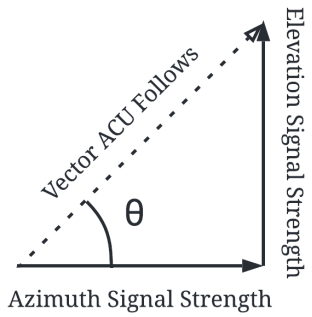

Fig. 5 Modified Step Tracking Mathematics

$$\theta = \arctan \frac{Elevation\ Signal\ Strength}{Azimuth\ Signal\ Strength} \tag{1}$$

The system takes the arc tangent of these two values to calculate the angle of greatest signal change, which is the direction the ACU antenna will move along. The modified algorithm then calculates a distance for the movement as proportional to the last signal change. Lastly, the function sends motor commands to move with the calculated angle and distance. If the signal strength between the starting and ending point is lower than a specified threshold, the function waits for a predefined time before continuing tracking to avoid unnecessary movement. Figure 6 illustrates the process of an ACU using this method to acquire a satellite connection.

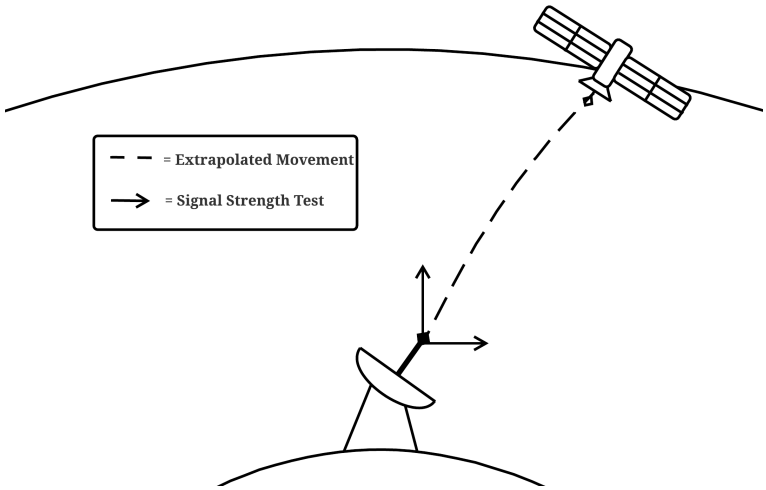

Fig. 6 Modified Step Tracking Function Testing Signal Strength Difference in Two Axes Before Extrapolating an Orientation Trajectory

This tracking method was created as a simplified version of step tracking, serving as a proof of the economical hardware. In addition, because a low final cost was a leading design goal, the modified step tracking approach was chosen over monopulse tracking and electronic beam steering due to their high-cost requirements [3] [11]. The developed tracking method is antenna agnostic, meaning it can accept any type of antenna under the weight limit and can operate with any Radio Frequency (RF) chain as long as software level adjustments are made to accommodate the changes. This was imperative to the rapid prototyping and testing of the ACU, and a photo of the specific setup used for testing can be seen in Figure 7

Fig. 7 hlA Photo of the Testing Setup for the ACU.

For future research, this novel step tracking approach could be further studied to establish degree accuracy and acquisition speed, but it fulfills its function of validating the hardware.

III. Mechanical System

A. Motor Selection

The driving requirements are:

1) The minimum elevation angle for the user terminals shall be 15°.
2) The tracking speed shall be three degrees per second, which can track LEO and MEO orbits.
3) The minimum tracking step size shall be 0.5°.
4) The antenna must be positioned 360° azimuth down to a 15° elevation

The system was designed to accommodate a frequency of up to 30 GHz with a 1.5 m antenna; however, the system should be able to accommodate higher frequencies with smaller antennas. To achieve these requirements, bevel gears were used to rotate the antenna vertically and horizontally and a motor that could meet these specifications was chosen.

The ACU utilizes an antenna that is 60 cm in diameter, which gives the ACU a frame height greater than 30 cm and minimum torque of 10 Nm. This torque value comes from the system requirement that the antenna is 6.8 kg with a moment arm of 15 cm. This means that 10 Nm is the minimum torque and a factor of safety of two gives a value of 20 Nm for the torque. Additionally, the antenna has a 36.67° beam width, which means that the tracking step size must be 0.5°. These requirements are listed in Table 1.

Table 1 Motor Requirements

System Requirement	Motor Requirement
Antenna is 6.8 kg. Moment arm is 15 cm.	Minimum torque of 20 Nm
Antenna has 0.5 dB beamwidth	Tracking step size: 0.5°

For this system, a NEMA 23 Stepper Motor was chosen. This motor has a gear ratio of 47:1, which is a high enough gear ratio to meet the required tracking speed but still have a high maximum torque. This motor was chosen because it has low speed and high torque with a maximum torque of 40 Nm, which satisfies the requirements. Additionally, the step size of this motor is 0.46° below the required step size at 0.039°, which provides an adequate level of tracking precision. The price of these motors, $53.50, helped minimize the cost of the ACU as it prevented the need to purchase an electronically steered antenna.

B. Antenna Mount and Midsection

The antenna is mounted to another U-shaped frame that fits around the outer shell onto the tilt shaft. This antenna mount connects to the tilt shaft via shaft hubs and mounting pins on either end, which can be seen in Figure 8. The tilt shaft (12 mm diameter) is driven by 90° miter gears held in place via plastic shaft spacers. The tilt shaft turns on two pillow blocks, each mounted to the main body via two 16 mm M5 bolts.

Similar to the antenna mount, the midsection consists of a folded aluminum sheet metal assembly of 0.41 cm in thickness. The midsection consists of two sub-sections: an upper U-frame that provides tilt functionality and connects to the tilt shaft, and a lower box frame that provides pan functionality as featured in Figure 8. The lower box frame attaches to the ground mount pole.

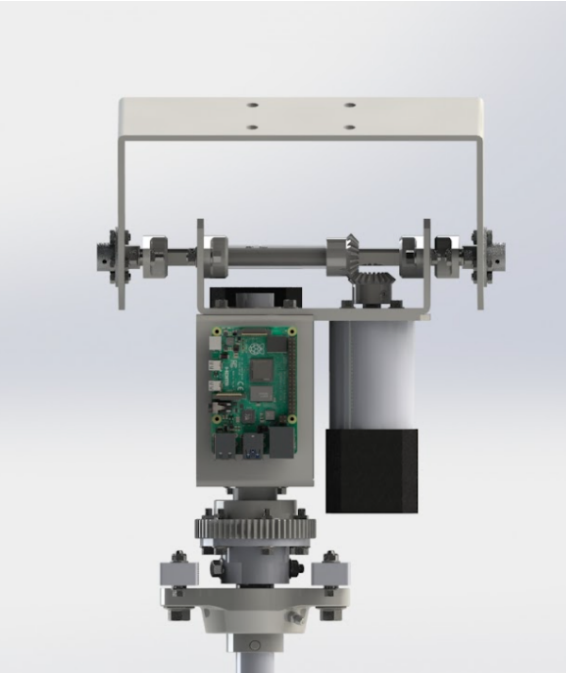

Fig. 8 Rendering of Mechanical System

This shaft, when rotated, moves the antenna through the shaft hubs on the antenna mount. The main body is attached to the bottom section via four ¼-20 bolts. Finally, the main body has two sets of mounting holes for the tilt motor and a limit switch.

C. Ground Mounting

The assembly is supported by a 25 mm pole that can either be buried in the ground or mounted to the side of a house, similar to how other companies mount their antennas. There is a 25 mm pillow block that attaches to the bottom of the plastic shell. This, and the two pan bearings, allow the assembly to rotate 360° around the ground pole for full functionality. The pillow block is attached to two mounting blocks that must be installed before shell halves are connected. Mounting blocks have M6 threads for pillow block attachment and mount in place via two 25 mm M5 bolts.

IV. Electrical Controls Subsystem

The electrical system of the ACU operates as the decision-making and power unit: providing power, direction, and speed to the motors. A 24 Volt (V) 5 Amp (A) power supply was used for the entire system. The Raspberry Pi functions as the brains of the unit while the Pluto SDR reads the current received signal strength. Alternatively, any microcontroller that can output 3.3 V logic could have been used, including an Arduino Uno. In this scenario, a Raspberry Pi was used due to its faster compute speeds, ability to multitask, and connect wirelessly to the internet.

In order to update the software running on the Raspberry Pi, methods including the wireless method of Secure Shell Protocol (SSH) or directly plugging into the unit can be used. The SSH method is preferred as accessing the Ethernet port is physically difficult. The Pluto SDR needs to have access to the signal exiting the Low Noise Blocker (LNB) and a splitter will have to be used if modem access is required. The SDR connects through the standard Universal Serial Bus (USB) to the Raspberry Pi.

The ACU system is antenna agnostic, meaning it works with any provided antenna and LNB that due not exceed weight limitations. The LNB and power inserter connect to the SDR via a provided coaxial cable. The configuration between the antenna system and the SDR is handled through the Raspberry Pi. As discussed earlier, two brushless DC bipolar stepper motors were used to handle the pan-tilt mechanism. Clock-in and serial controlled stepper motor drivers were used as well.

V. Software

This section provides an overview of the non-active coded functions used in the tracking algorithm. These functions complemented the open loop modified step tracking algorithm by simplifying initial acquisition. The sweep function initially acquires a satellite by performing a 360° sweep across multiple elevations. The scheduler function saves a satellite's flight path, allowing the ACU to search along this path for faster future acquisition. Both are preprogrammed functions that allow the ACU to acquire a satellite before active tracking takes over by the modified step tracking algorithm.

A. Signal Sweep

The signal sweep is classified as a programmed steering function responsible for initially finding a satellite without outside position information [3]. The alpha prototype does this by scanning the sky from 87° vertical down to horizontal while spinning 360°. As the ACU sweeps, it checks the signal strength at every degree in the azimuth direction and then shifts to incrementally higher elevations. The signal sweep function was written to only scan a small bandwidth around a preset center frequency, which was set to target the beacon frequency of the satellite(s). In the case there are more than one satellites with the same beacon frequency, the ACU would not be able to distinguish between them. The ability for the ACU to read incoming signal and identify the transmitting satellite can be added, but this would require decrypting the signal which was not possible within the scope of this research. This sweep assures that the entire horizon is swept up to nearly vertical, ensuring the satellite is acquired. Figure 9 illustrates a heat map of signal strength created during a sample scan for the Galaxy-19 geostationary (GSO) satellite.

The ACU sweeps to observe a Signal-to-Noise Ratio (dB) that surpasses a user-defined threshold, indicating the target satellite is within view. Once the signal strength threshold is surpassed, the signal signature is verified and the ACU will exit signal sweeping and transfer to the modified active step tracking technique. In short, the signal sweep function serves as an aid to the active tracking algorithm by initially locating the satellite at startup.

B. Scheduler

A scheduler programmed steering function was implemented to aid the ACU in the faster acquisition of a satellite that travels over a repetitive arc and serves as an alternative to the signal sweep function. Figure 10 exhibits the ACU immediately tracking along a predetermined trajectory rather than utilizing the sweep function.

This is known as an orbit determination technique where an orbit estimator analyzes previous orbits and then determines a best-fit orbit for the ACU to track [3]. Previous research notes the benefit of coupling an orbital estimator with an auto-tracking system as having "greatly enhance[d] the achievable tracking performance" [3]. The scheduler function assumes there will be another satellite moving on the same arc as the previously tracked satellite. The function records the movements made during the active tracking, but only after initial acquisition. This means the ACU ideally only tracks the satellite's arc and not extraneous movements. In relation to the signal sweep and modified active tracking functions, the scheduler functions as a shortcut if the ACU is going to be tracking one satellite that follows a repetitive

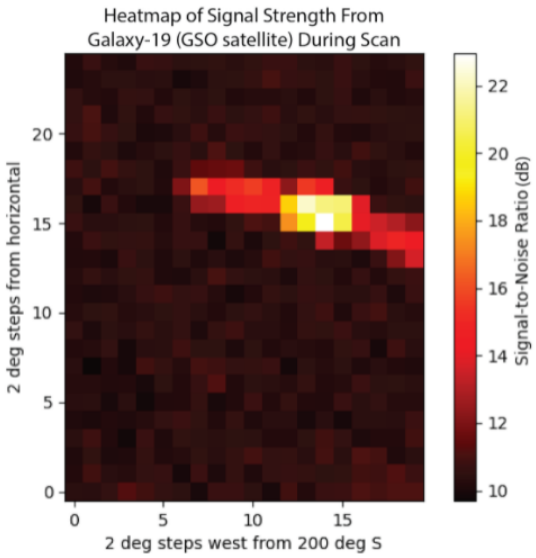

Fig. 9 A Heat Map of a Sample Scan for the Galaxy-19 GSO Satellite

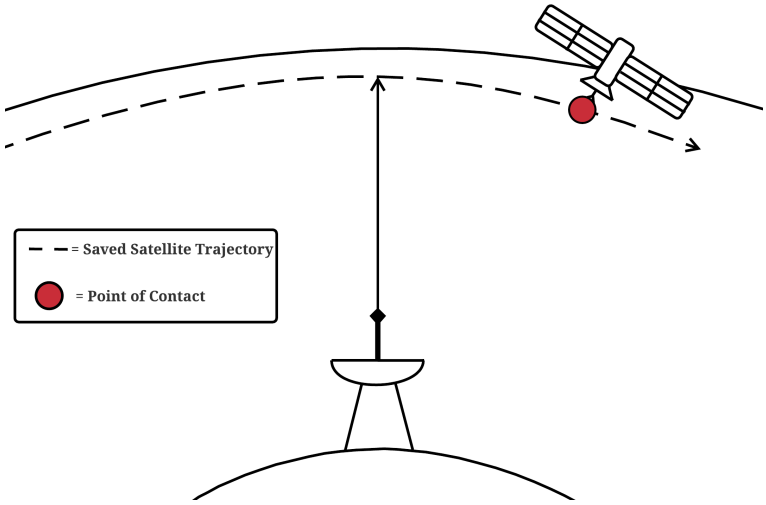

Fig. 10 Scheduler Function Moving the Receiver along the Saved Trajectory Path of the Satellite

arc, avoiding the more intensive scan sweep process to acquire a satellite before handing off to the active tracking algorithm.

VI. Current Prototype

The current alpha prototype of the ACU uses manual tracking from user input, off-the-shelf electrical and mechanical systems, and a simple pan-tilt mechanism for positioning the antenna. This system is used to verify and test the design and is shown in Figure 11.

Fig. 11 Fully Integrated Alpha Prototype of the ACU

For this paper, the prototype contained a Ku-band antenna setup with a Raspberry Pi connected to an SDR, which is used to read the signals and is connected as seen in Figure 12.

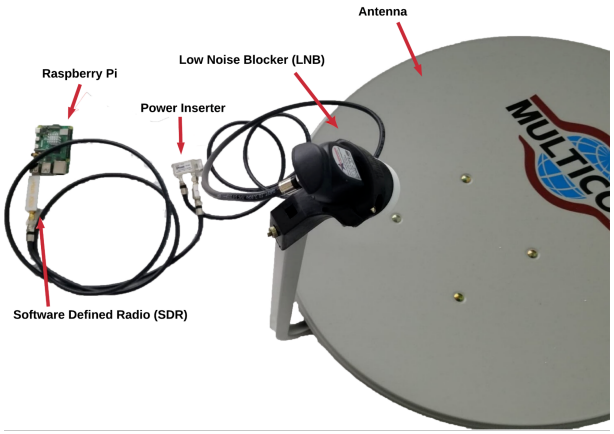

Fig. 12 Annotated Photo of the Dish-to-Raspberry Pi Setup

This is then mounted on a simple pan-tilt mechanism driven by two bipolar stepper motors and attached to a tripod as shown in Figure 11.

VII. Conclusion and Next Steps

The alpha prototype developed was used as a test bed for the different electrical and mechanical features of the system. All individual components were tested and the system successfully tracked and received data from LEO and MEO satellites. Throughout the design and test process, there were actions that could have been taken to improve the ACU system's performance and cost, the main of which being in the mechanical subsystem. The mechanical subsystem will need to be further cost optimized in the sourcing of the gears and the motors, while the motors themselves will be reoriented to make a more compact design with less mechanical wear. The cost of these components can be expected to fall with economies of scale. On the electrical side, better cable management techniques will be implemented by grouping cables together and attaching them to the main body to reduce strain and points of failure. On the software side, the modified step tracking algorithm can be further developed beyond its proof of concept phase and its performance compared to established tracking algorithms. In addition, a hybrid approach could be implemented to utilize both step tracking and TLE for satellite tracking, potentially improving accuracy and resilience. All of the improvements will be evaluated and a second "beta" prototype will be developed to further lower the cost while optimizing the system to ensure robust performance. The beta prototype will also undergo long-term testing that will extend up to a month, to determine failure modes and locate points of high mechanical wear. The beta prototype development, implementation, testing, and lessons learned will be shared in a future publication.

Acknowledgements

The antenna control unit research and development was part of the Senior Capstone Project at Olin College by Kyle Emmi, Brandon Zhang, Victoria Wyatt, Jillian MacGregor, Sebastian Calvo, and advised by Scott Hersey. The project was mentored by Mangata Network's chief architect Ken Mentasti and Christian Rodriguez. This paper was written by students at the OSSTP Group that will continue to iterate and develop the ACU for a variety of use cases.

Financial disclosure

The project was funded by the National Science Foundation (NSF) Spectrum Planning Grant, Award Number 2132700.

Conflict of interest

The authors declare no potential conflict of interests.

References

[1] N. Pachler, E. F. C., I. del Portillo, and Cameron, B. G., ""An Updated Comparison of Four Low Earth Orbit Satellite Constellation Systems to Provide Global Broadband"," *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–7. https://doi.org/10.1109/ICCWorkshops50388.2021.9473799.

[2] Mulla, A. A., and Vasambekar, P. N., "Overview on the development and applications of antenna control systems," *Annual Reviews in Control*, Vol. 41, Jan 2016, pp. 47–57. https://doi.org/10.1016/j.arcontrol.2016.04.012.

[3] G. J. Hawkins, D. J. E., and McGeehan, J. P., "Tracking systems for satellite communications," *IEE Proceedings F (Communications, Radar and Signal Processing)*, Vol. 135, No. 5, Oct 1988, pp. 393–407. https://doi.org/10.1049/ip-f-1.1988.0047.

[4] Norman, T., "12 - Radio Frequency Systems," *Integrated Security Systems Design (Second Edition)*, 2014, pp. 251–266. https://doi.org/10.1016/B978-0-12-800022-9.00012-7.

[5] B. Rama Rao, L. P. C., and Davis, R. J., "SHF Cassegrain antenna with electronic beam squint tracking for high data rate mobile satellite communication systems," *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, Vol. 2, June 1994, pp. 1028–1031. https://doi.org/10.1109/APS.1994.407915.

[6] M. N. Thejaswini, M. R. M., and Yesobu, B., "Intelligent Satellite Tracking for Antenna Control System," 2012.

[7] M. M. Alam, M. M. I., and Mansoor, A. T., "Fully automated satellite tracking system for directional antenna," 2018. https://doi.org/http://dspace.bracu.ac.bd/xmlui/handle/10361/10932.

[8] Chen, C., and Wang, Y., "Research of the Variable Step Size Algorithm of Antenna Tracking Satellite for SATCOM On-the-Move," *5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol. 2, August 2013, pp. 490–493. https://doi.org/10.1109/IHMSC.2013.264.

[9] CG. Han, W. W., B. Du, and Yang, B., "A Novel Hybrid Phased Array Antenna for Satellite Communication on-the-Move in Ku-band," *IEEE Transactions on Antennas and Propagation*, Vol. 63, No. 4, April 2015, pp. 1375–1383. https://doi.org/10.1109/TAP.2015.2389951.

[10] F. Tiezzi, J. P., and Vigano, C., "S-band transmit/receive antenna with electronically switched beams for mobile satellite systems," *6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC)*, Sep 2012, pp. 62–67. https://doi.org/110.1109/ASMS-SPSC.2012.6333107.

[11] S. Vaccaro, J. P., D. L. del Río, and Baggen, R., "Low cost Ku-band electronic steerable array antenna for mobile satellite communications," *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, April 2011, pp. 2362–2366.

[12] M. Nasrollahnejad, Y. M., S. Nowdeh, and Moharlooei, P., "LEO Satellite Tracking Using Monopulse," *Middle-East Journal of Scientific Research*, Vol. 11, Jan 2012, pp. 723–726.

Part II

Estimating the Probability That the Explosion of an Ink Sphere Produces a Dictionary

Estimating the Probability That the Explosion of an Ink Sphere Produces a Dictionary

Braden Oh*
Olin College of Engineering, Needham, MA, 02492

Andrew Haviland†
Brigham Young University, Provo, UT 84602

Michael S. Rugh, Ph.D.‡
Texas A&M University, College Station, TX 77843

There is a thought experiment that compares the probability of life originating by random chance to the probability of a dictionary being printed in the explosion of a printing shop; this thought experiment has not been formally analyzed. This paper presents a highly simplified abstraction of the thought experiment and calculates the probability for the likelihood that an explosion of a sphere of ink results in a spray pattern that prints a 1989 Merriam-Webster English dictionary. The model presented consists of a sphere of ink exploding within a spherical shell composed of pieces of printing paper. The ink sphere is modeled as a composite of rectangular pyramid sectors that expand outward until their footprint perfectly overlaps a target piece of paper. Each sector is modeled as a stack of discrete layers of ink droplets, allowing the ink droplets within each sector to be quantified by a three-dimensional Riemann sum. The possible arrangements of these droplets are modeled by combinations and permutations that require the evaluation of enormous factorials to solve (e.g., 5,025,780!). These factorials are too large for a computer to evaluate by multiplication—or even Sterling's Approximation—due to limitations in compute time and decimal precision. As a result, an algorithm for using floating point numbers to conduct large combinatorial approximations in logarithmic space is derived and presented. The computed likelihood of a dictionary being printed within this model is $\approx 1.30 \times 10^{-817,692,555}$, or, in a more familiar form, ≈ 1 in $7.68 \times 10^{817,692,554}$.

I. Introduction

THERE is a thought experiment that compares the probability of life originating by random chance to the probability of a dictionary being printed in the explosion of a printing shop. It is unclear who first posed this thought experiment. Academic analysis has been given to the probability of life as we know it arising [1–3]. Academic analysis has also been given to determine the probability of extremely low-likelihood events, such as the probability of a person quantum tunneling through a wall, the probability of infinite monkeys with typewriters producing the complete works of Shakespeare, or the probability of a fully intact brain appearing out of quantum fluctuations in the vacuum of space [4–7]. Academic literature studying the thought experiment of an explosion printing a dictionary, however, is not readily available. Although difficult to model the entire scenario of a print factory exploding, this paper explores a highly abstracted and idealized model with the purpose of providing an entry point for future academic exploration.

II. Background and Assumed Model

The model used in this paper consists of a sphere of ink exploding within a spherical shell comprised of pieces of printing paper; the distribution of ink particles is modeled by sectors of the sphere that expand outwards, eventually striking the walls of the paper shell. A diagram of this model is shown in Fig. 1. In order to analyze this model mathematically, many simplifications must be made, including the following primary assumptions:

*Engineering Physics
†Mathematics
‡Research Scientist, Learning Interactive Visualization Experience (LIVE) Lab

1) The ink sphere contains exactly the amount of ink necessary to print a single dictionary.
2) The roughly-spherical paper shell is comprised of the number of rectangular pieces of paper necessary to print a single dictionary.
3) The pieces of paper comprising the shell are at fixed locations in space.
4) All pages of the model dictionary contain the average number of pixels per page of a Merriam-Webster *Webster's Dictionary of English Usage*.
5) The dictionary is printed at a standard print resolution of 300 dots per inch (dpi).
6) At the instant of explosion, the ink sphere atomizes into identical droplets, each with the diameter to produce a 300 dpi pixel.
7) The sphere expands uniformly, resulting in an equal number of ink droplets per sector.
8) The sphere is analyzed as a set of identical, expanding sectors, with the number of sectors being equal to the number of pages in the dictionary and each containing enough ink to print one page.
9) Each expanded sector's footprint encloses a single piece of paper.
10) Each sector is a rectangular pyramid comprised of a stack of flat layers of ink droplets.
11) Ink droplets within any single layer are randomly arranged within the target footprint, but no two droplets in a single layer can land in the same location.
12) The order of the printed pages does not matter.

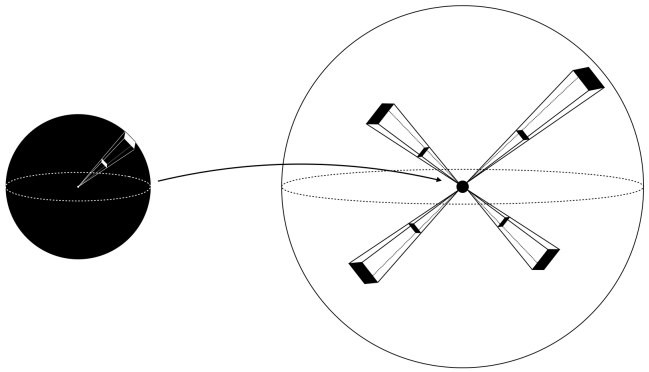

Fig. 1 The model this paper uses is a sphere of ink comprised of sectors that are approximated by rectangular pyramids. When the explosion occurs, these sectors expand outward until they strike a roughly spherical shell comprised of pieces of paper. The footprint of each pyramid encloses a single piece of paper, ensuring the ink from that sector lands somewhere on that page.

The realistic nature of these assumptions is questionable but acceptable within reason for a highly approximated model. Although it is not possible to create a perfectly closed shell (polyhedron) with identical, rectangular faces, prior work in hemisphere partitioning has shown that it is possible to partition a hemisphere into equal areas with constrained aspect ratios that well approximate same-size rectangles [8]. Although the likelihood of an explosion atomizing ink droplets instantly and identically is low, the concept of flash boiling for ideal spray generation has been researched and has a potential application in the automotive engine industry for the purpose of optimizing fuel injection [9, 10]. Although dot-matrix printers can produce resolutions as low as 60 dpi, such printers are no longer commonly used, so a resolution of 300 dpi (a typical medium to high quality resolution) is used by the model [11, 12].

Although this model does not analyze an entire print factory, it does allow for quantitative analysis of the probability that a random explosion of ink particles within an idealized environment results in a Merriam-Webster English dictionary; it thus serves as an entry point for further academic analysis and discussion of this thought experiment.

III. Numbers of Pixels

The dictionary used in this model is the 1989 edition of the Merriam-Webster *Webster's Dictionary of English Usage*. To determine the average number of pixels per page (as required by assumption 4) 30 pages were selected at random from the 973 pages that make up the main body of this edition (ignoring brief preface and suffix sections, such

as the edition notes and bibliography). These 30 pages were extracted from a portable document format (PDF) file of the dictionary, converted into black and white bitmaps, and had their pixels counted by a Python script. Of 149,261,354 total pixels counted, 19,476,801 were black, yielding an average of 13.0488% black pixels per page.

The dimensions of a page of the 1989 Merriam-Webster English dictionary are approximately 6.15×9.08 inches. At a resolution of 300^2 dots per square inch, each page is comprised of 5,025,780 pixels. Taking 13.05% of these pixels yields 655,803 black pixels (resulting in 638,096,319 total ink droplets necessary to print the entire dictionary). The average number of pixels per page will be denoted hereafter as $\Sigma N = 655{,}803$ droplets.

IV. Ink Droplet Physical Properties

By primary assumption 5 the dictionary is printed at a resolution of 300 dpi. By primary assumption 6 the ink sphere atomizes into identical droplets, each capable of yielding a pixel that measures $\frac{1}{300}$ of an inch, or $84.\overline{6}$ microns. Prior research in the microscopic topography of ink on paper has found that a typical layer of ink on a paper is approximately 2.5 microns deep [13]. Considering each pixel to be a cylinder with a diameter of $84.\overline{6}$ microns and a thickness of 2.5 micrometers, we can calculate the volume of ink in each pixel. This volume, in turn, equals the volume of ink in a spherical droplet of the ink spray, $V_d = 14\,075.21$ micron^3, which will have a diameter of $D_d = 29.96$ microns. The total volume of ink necessary to print a dictionary is therefore $V_d \times \Sigma N = 8.98$ cm^3. Before being exploded, this volume of ink occupies a sphere of radius $r_0 = 1.29$ cm.

V. Geometry of an Expanding Sector

A sector of the initial ink sphere is approximated by a rectangular pyramid that, by assumption 9, has an expanded footprint that covers one page of the dictionary (the entire initial ink sphere is comprised of 973 pyramids, one for each page of the dictionary). This is similar to the analysis of the footprint of a sensor with a rectangular field of vision, see [14]. By assumption 10, the rectangular pyramid is a composite of layers of ink particles, each with a distinct thickness of D_d. The pyramid's volume can thus be considered as the sum of the volumes of thin boxes, each of which has a rectangular base and a height of D_d; this is a volumetric Riemann sum similar to the one used to derive the formula for the volume of a pyramid for students of calculus [15, 16]. A diagram of this geometry and relevant dimensions is shown in Fig. 2.

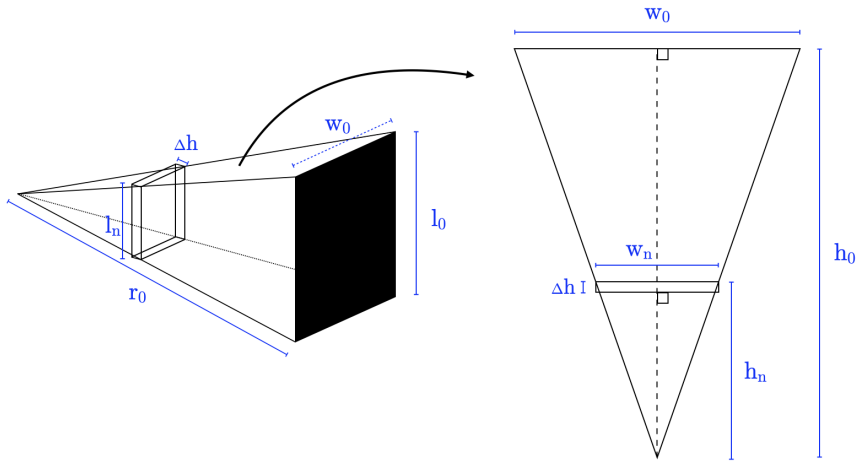

Fig. 2 Each pyramidal sector is represented as a composition of slices, with each slice being the thickness of an ink droplet and containing a discrete number of ink particles. Left is a sector diagram with a discrete slice shown, and right is a top-view of the same geometry. The labeled values are used in the following set of calculations. w_0 **and** l_0 **are the dimensions of a page, 6.15 and 9.08 inches (156,210 and 230,632 microns), respectively, and** $\Delta h = D_d$**.**

The height of the expanded pyramid, or the distance from the center of the initial ink sphere to the center of the target piece of paper, is $h_0 = \sqrt{r_0^2 - (\frac{w_0}{2})^2}$, a function of the radius of the spherical paper shell, r_0, and the width of the

target piece of paper, w_0. When taking a Riemann sum of the pyramid's volume along the h axis, the volume component at a particular step, n, is a rectangular prism with length l_n, width w_n, and thickness Δh (which is equal to the diameter of a single droplet of ink, D_d). The values of l_n and w_n can be found with similarity of triangles by multiplying l_0 and w_0 each by the ratio of the altitude of the pyramid at that step, h_n, to the expanded altitude, h_0. The Riemann sum approximating the total volume of the pyramid, ΣV, is the sum of all volume components:

$$\Sigma V = \sum_{n=0}^{m} \frac{w_0 \times l_0 \times \Delta h^3 \times n^2}{r_0^2 - (\frac{w_0}{2})^2} \tag{1}$$

The limit of the sum, m, is the integer number of layers in the pyramid rounded to the nearest integer: $m = \frac{h_0}{\Delta h}$. Assumption 10 models the ink in a sector as a series of layers comprised of ink droplets, and assumptions 1, 4, 7, and 8 establish that each sector contains exactly the number of droplets necessary to print a single page. The number of droplets in a particular layer, $N(n)$, is therefore given by

$$N(n) = \frac{V(n)}{\Sigma V} \Sigma N \tag{2}$$

where $V(n)$ is the volume of that layer of the sector,

$$V(n) = \frac{w_0 \times l_0 \times \Delta h^3 \times n^2}{r_0^2 - (\frac{w_0}{2})^2} \tag{3}$$

The parameters for evaluation of these expressions (applied to the initial, un-exploded ink sphere) are the following:

$$\begin{aligned} w_0 &= 15.62 \text{ cm} \\ l_0 &= 23.06 \text{ cm} \\ \Delta h &= 29.96 \text{ microns} \\ n &= 430 \text{ layers} \\ r_0 &= 1.29 \text{ cm} \end{aligned}$$

A. Computational Evaluation of $N(n)$, with Corrections

Equations 2 and 3 can be used to generate a set of values that denote the number of ink particles at each layer of the sector, $\{N(0), N(1), N(2), ...N(m)\}$. This set was calculated computationally in Python 3.8.5 and resulted in 655,795 total droplets across all layers. ΣN = 649,226 droplets, however, meaning the Riemann sum calculated an extraneous 6,569 droplets (an error of +1.01%). To correct for this error, each layer had subtracted from it a portion of these extraneous droplets proportional to the fraction of the total number of sector droplets contained within that layer. This resulted in the subtraction of 6,566 extraneous droplets and reduced the error in the total number of droplets to below +0.0005%.

VI. Arrangements Within a Single Layer

By assumption 11, the ink droplets are randomly distributed throughout the target area; because all ink droplets are identical, the number of arrangements for droplets in a single layer can be modeled as a simple combination,

$$C(n) = \binom{L}{N(n)} \tag{4}$$

where L is the total number of locations that an ink droplet can land, $N(n)$ is the number of droplets in a given layer, and $C(n)$ is the number of possible arrangements for the particles in that layer. In Section III, it was determined that a resolution of 300^2 dots per square inch yields 5,025,780 total pixels per page (black or white), thus L = 5,025,780 locations. This function can be used to generate a set of values that denote the number of arrangements for each layer of the sector, $\{C(0), C(1), C(2), ...C(m)\}$. Also by assumption 11, although no two ink particles in a single layer can land in the same location, ink particles in subsequent layers *can*. Thus, the product of all items within the set of C values is

the total number of possible arrangements for all ink particles in an entire sector. This quantity (hereafter referred to as Γ) can be represented in product notation as

$$\Gamma = \prod_{n=0}^{m} \binom{L}{N(n)} \tag{5}$$

where m continues to represent the number of layers in the sector.

VII. Accounting for Multiple Correct Arrangements

Let us consider a page to be "successfully printed" when a particular subset of ΣN cells are filled with ink. Although there is exactly one subset of cells that results in a correctly printed page, those ΣN droplets of ink may be arranged in any manner within those cells. This repetition is illustrated in Fig. 3 with a simple example of five ink droplets filling a single configuration of five target cells.

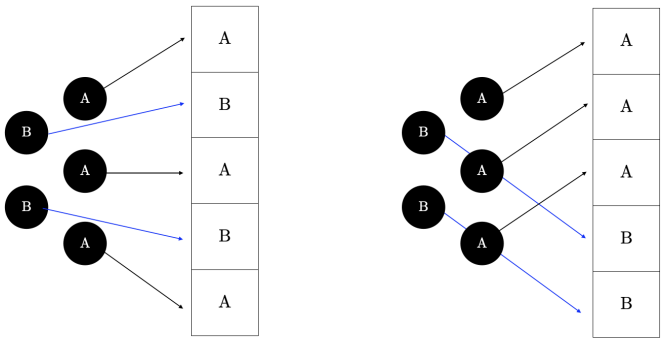

Fig. 3 A simplified example of printing the letter "l" where five ink droplets are arranged into two layers, A and B. To successfully "print" this letter, all five target cells must be filled. In both the left and right scenarios all five cells are successfully filled, but both solutions are counted even though they both produce the same (single) outcome. To determine the number of correct arrangements, a permutation with repetition must be employed.

In Fig. 3, five droplets are arranged into two layers, with three droplets in A and two in B. Two possible arrangements of these five droplets are shown in Fig. 3. Both of the arrangements shown generate a "correctly printed" page despite being arranged differently. The total number of *correct* arrangements of these five particles (accounting for the fact that two particles within the same layer are identical) is described mathematically as a permutation with repetition, $\frac{5!}{3!\times 2!}$. This principle is applied to the sector as a whole by the following permutation with repetition:

$$\frac{\Sigma N!}{\prod_{n=0}^{m} N(n)!} \tag{6}$$

The actual probability of a page being printed is the ratio between the number of correct arrangements and the total number of arrangements. Γ provides the total number of arrangements; the number of correct arrangements is given by Expression 6. The probability ratio, Λ is thereby expressed as

$$\Lambda = \frac{\Sigma N!}{\Gamma \prod_{n=0}^{m} N(n)!} \tag{7}$$

VIII. Computing Enormous Factorials

In order to evaluate Λ, factorials of L, ΣN, and various large $N(n)$ values must be computed. The largest factorial required is $L!$, which requires the computation of 5,025,780!. Exact computation of large numbers can take enormous

amounts of computing time; a study of the number of legal moves in games of Go required 8000 central processing unit (CPU) hours to generate an exact number with an order of magnitude of 136 (for reference, 5,025,780! has an order of magnitude of 10^7) [17]. One commonly used function for calculating enormous factorials is Sterling's Approximation [18, 19],

$$L! \sim \sqrt{2\pi L}\left(\frac{L}{e}\right)^L \tag{8}$$

Although less computationally intensive than repeated multiplication, Sterling's Approximation for $L!$ requires the evaluation of $(\frac{L}{e})^L$, an operation that still requires a large amount of computational time. Furthermore, the resulting number (which also has an order of magnitude of 10^7) is too large to be stored by a conventional data type. Python 3.8.5 floating-point numbers are almost always mapped to IEEE-754 "double precision" [20]. The maximum value that Python can represent has an order of magnitude of 308, much too small to handle $(\frac{L}{e})^L$, and so Sterling's Approximation will not suffice for this computation. We can, however, effectively utilize Python's 17-digit floating point precision by exploiting the fact that multiplication in decimal space is equivalent to addition in logarithmic space. Thus,

$$L! = \prod_{n=0}^{L-1}(L-n) \tag{9}$$

$$\log_{10} L! = \sum_{n=0}^{L-1}\log_{10}(L-n) \tag{10}$$

$L = 5,025,780$, so its logarithmic representation is $\log_{10}(5,025,780) \approx 6.70$, which is a number easily handled by a floating point number. When evaluated in Python 3.8.5, the sum in Equation 10 evaluates to the following:

$$\log_{10} L! \approx 31,496,109.622662853 \tag{11}$$

$$\therefore L! \approx 10^{31,496,109.622662853} \tag{12}$$

The power of ten in Equation 12 can be factored into two terms: 10 raised to the integer portion of the power and 10 raised to the decimal portion of the power. 10 raised to the decimal portion can be easily evaluated by a computer, while 10 raised to the integer portion provides an order of magnitude for the computed decimal. This algorithm is illustrated by factoring the output of Equation 12:

$$L! \approx 10^{0.622662853} \times 10^{31,496,109} \tag{13}$$

$$\approx 4.19 \times 10^{31,496,109} \tag{14}$$

Being that combinations are a function of factorials, this technique for computing large factorials can be used to create approximations for every factorial necessary to evaluate Λ.

IX. The Probability of an Explosion Printing a Dictionary

With this technique for computing large factorials, the effective probability ratio, Λ_{ef}, that a single page of the dictionary is printed can be directly evaluated as expressed in Equation 7. Thus,

$$\Lambda_{ef} \approx 1.78 \times 10^{-840,386}$$

However, being that a page can be printed either right side up or upside down and still be a recognizable page (and arbitrary rotations of the printed sphere are ignored par assumption 3), the actual probability ratio is twice Λ_{ef}, hence,

$$\Lambda \approx 3.56 \times 10^{-840,386}$$

This ratio can be translated into a more familiar form by taking its reciprocal; thus, the chance of a single page of the dictionary being successfully printed is 1 in $2.81 \times 10^{840,385}$. Raising Λ to the power of 973 (the total number of pages in the dictionary) will provide the probability that all pages are successfully printed. However, this power includes all possible arrangements of those pages. By assumption 12, the order of the printed pages does not matter, so the final probability ratio is given by Λ^{973} multiplied by the total number of arrangements of all printed pages. The total number

of arrangements of all pages is given by the permutation ${}_{973}P_{973}$, as 973 positions can hold 973 unique pages. This permutation simplifies to 973!, and thus the final probability ratio, Ω, that a dictionary is printed is

$$\Omega \approx 973! \times \Lambda^{973} \tag{15}$$

$$\approx 1.30 \times 10^{-817,692,555} \tag{16}$$

Taking the reciprocal of Ω yields a more familiar form: 1 in $7.68 \times 10^{817,692,554}$.

X. Conclusions

The printing factory explosion thought experiment, up until this point, had not been formally evaluated. Although the original scenario—that of an explosion in a printing factory—is extremely complex to model in full, a highly simplified model consisting of an explosion of a single sphere of ink within a static paper shell can be mathematically modeled and estimated. Within our model, the probability of a dictionary being printed is found to be 1 in $7.68 \times 10^{817,692,554}$.

This probability can be compared to the probability of certain events described in popular media and other thought experiments. Students at the University of Leicester performed an investigation into the likelihood that a person moving at $0.99c$ could quantum tunnel through a solid wall, finding the likelihood of such an event to be $10^{-10^{28}}$ [4]. On the other hand, the probability of a monkey randomly typing the complete works of Shakespeare has been estimated at about $10^{-10^{7.15}}$ [5]. We calculated the probability of an explosion printing a dictionary under our model to be approximately $10^{-10^{9}}$, which is much higher than the probability of a person quantum tunneling through a solid wall and much lower than the probability of a monkey randomly typing the entire works of Shakespeare.

We present this as our estimate for the probability of a dictionary being printed in the circumstances of this model. It is important to note, however, that this does not establish an estimate for the probability of the entire print factory explosion thought experiment, as the sheer scale of an explosion in a printing factory introduces factors that could drive the likelihood of a dictionary being printed either up or down from that of our model. For instance, a print factory could be modeled as a series of many explosions identical to this one, effectively driving the likelihood up. Furthermore, this model assumes that the ink droplets must fill a specific arrangement perfectly to produce a dictionary, whereas it is much more likely to produce a dictionary that has almost all droplets in the correct arrangement, with some out of place; setting thresholds that accept these imperfect dictionaries would also drive the likelihood up. Conversely, adding complexity to the atomization of ink droplets or assuming non-uniform ink distribution throughout the spray would introduce variables that could supply excessive or insufficient ink to particular pages, effectively driving the likelihood of success down. This model thus serves as an entry point for more complex analyses of the thought experiment, opening the door for the evaluation of higher-fidelity models that can establish a more accurate estimate of the likelihood that a dictionary is printed in the explosion of a printing factory.

References

[1] Loeb, A., Batista, R. A., and Sloan, D., "Relative likelihood for life as a function of cosmic time," *Journal of Cosmology and Astroparticle Physics*, Vol. 2016, No. 08, 2016, p. 040–040. https://doi.org/10.1088/1475-7516/2016/08/040, URL http://dx.doi.org/10.1088/1475-7516/2016/08/040.

[2] Kipping, D., "An objective Bayesian analysis of life's early start and our late arrival," *Proceedings of the National Academy of Sciences*, Vol. 117, No. 22, 2020, pp. 11995–12003. https://doi.org/10.1073/pnas.1921655117, URL https://www.pnas.org/content/117/22/11995.

[3] Scharf, C., and Cronin, L., "Quantifying the origins of life on a planetary scale," *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 113, No. 29, 2016. https://doi.org/10.1073/pnas.1523233113.

[4] Alam, A., Campbell, J., Phelan, J., and Warne, B., "The Flash and Quantum Tunnelling," *Journal of Physics Special Topics*, Vol. 15, No. 1, 2016.

[5] Weaver, E. R., "An exercise in probability," *Journal of the Washington Academy of Sciences*, 1965.

[6] Kittel, C., and Kroemer, H., *Thermal Physics*, W. H. Freeman and Company, 1980.

[7] Page, D., "Return of the Boltzmann Brains," *Physical Review D*, Vol. 78, 2006. https://doi.org/10.1103/PhysRevD.78.063536.

[8] Beckers, B., and Beckers, P., "A general rule for disk and hemisphere partition into equal-area cells," *Computational Geometry*, Vol. 45, No. 7, 2012, pp. 275 – 283. https://doi.org/https://doi.org/10.1016/j.comgeo.2012.01.011, URL http://www.sciencedirect.com/science/article/pii/S0925772112000296.

[9] Sher, E., Bar-Kohany, T., and Rashkovan, A., "Flash-boiling atomization," *Progress in Energy and Combustion Science*, Vol. 34, No. 4, 2008, pp. 417–439.

[10] Xu, M., Zhang, Y., Zeng, W., Zhang, G., and Zhang, M., "Flash Boiling: Easy and Better Way to Generate Ideal Sprays than the High Injection Pressure," *SAE International Journal of Fuels and Lubricants*, Vol. 6, 2013, pp. 137–148. https://doi.org/10.4271/2013-01-1614.

[11] Noronha, S. J., Basheer, S. Z., Vijay, M. N., Alnajjar, A., Sharma, B., and Singh, N., "A Comparative Study of Different Printed Documents to Estimate the Type of Printer Used," *Journal of Forensic Research*, 2017, pp. 1–7.

[12] Pappas, T. N., Dong, C.-K., and Neuhoff, D. L., "Measurement of printer parameters for model-based halftoning," *Journal of Electronic Imaging*, Vol. 2, No. 3, 1993, pp. 193 – 204. https://doi.org/10.1117/12.148574.

[13] Myllys, M., Häkkänen, H., Korppi-Tommola, J., Backfolk, K., Sirviö, P., and Timonen, J., "X-ray microtomography and laser ablation in the analysis of ink distribution in coated paper," *Journal of Applied Physics*, Vol. 117, No. 14, 2015, p. 144902. https://doi.org/10.1063/1.4916588.

[14] North, G. R., Valdes, J. B., Ha, E., and Shen, S. S. P., "The Ground-Truth Problem for Satellite Estimates of Rain Rate," *Journal of Atmospheric and Oceanic Technology*, 1994.

[15] Eriksson, K., Johnson, C., and Estep, D., *Double Integrals. In: Applied Mathematics: Body and Soul*, Springer, Berlin, Heidelberg, 2004.

[16] Hughes-Hallett, D., Gleason, A. M., and McCallum, W. G., *Calculus: Single Variable*, 7^{th} ed., John Wiley & Sons, Inc., 2017.

[17] Tromp, J., and Farneback, G., "Combinatorics of Go," *Lecture Notes in Computer Science*, Vol. 4630, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-75538-8_8#citeas.

[18] Diaconis, P., and Freedman, D., "An Elementary Proof of Stirling's Formula," *The American Mathematical Monthly*, Vol. 93, No. 2, 1986, pp. 123–125. URL http://www.jstor.org/stable/2322709.

[19] Eger, S., "Stirling's Approximation for Central Extended Binomial Coefficients," *The American Mathematical Monthly*, Vol. 121, No. 4, 2014, pp. 344–349.

[20] "Floating Point Arithmetic: Issues and Limitations," *Python 3.8.11 Documentation*, 2021.

Part III
Initial Testing of a 3D Printed Gas Diffuser for Hall Effect Thrusters

Initial Testing of a 3D Printed Gas Diffuser for Hall Effect Thrusters

Braden Oh*, James Jagielski†, Lily Dao‡, Ben Eisenbraun§, Avery Clowes¶, and Christopher Lee‖
Olin College of Engineering, Needham, MA, 02492

Albert Countryman**
Brandeis University, Waltham, MA, 02453

Christina Crochetiere††
Babson College, Babson Park, MA, 02457

The thrust of an electric propulsion engine is proportional to the number of ions ejected; ionizing higher percentages of propellant results in a higher thruster efficiency. Propellant ionization in a Hall thruster is optimized by reducing the mean free path of propellant particles. This can be achieved by reducing the axial velocity of the propellant. This team has designed and demonstrated a design for an azimuthal propellant diffuser that is unibody and cheap to additively manufacture. A diffuser was manufactured from Somos PerFORM and was tested in live-fire conditions. The diffuser successfully allowed the thruster to operate at steady state, though inspection of the diffuser post-firing showed evidence of uneven heating and cracking due to thermal stresses, indicating that greater thermal considerations must be made for future models of the diffuser.

I. Introduction

In the modern world, satellites are increasingly critical to advancing human progress. Satellites are a key technology for radio, television, GPS, and scientific measurements, such as weather monitoring and deep space science. They also power global knowledge-sharing and telecommunications capabilities of today, enabling us to monitor our natural world and be prepared to respond to challenges such as natural disasters and the effects of climate change. Another recent area of growth is satellite internet; although most internet infrastructure is currently established on Earth, satellite megaconstellations, such as those launched by OneWeb and SpaceX, are expanding the future of global communications.

In order to fulfill their roles, satellites must be placed in very specific orbits. Satellites are launched into orbit on combustion rockets that create high thrust for short periods of time. These rockets are very inefficient, however, so many modern satellites utilize electric propulsion once they reach orbit. Electric propulsion engines utilize electromagnetic fields to propel ions into space and can be five to ten times more efficient (per effective exhaust velocity) than liquid rockets [1]. Electric propulsion systems decrease the mass and volume cost required to operate satellites and are a major step being taken to improve space travel.

The Olin Plasma Engineering & Electric Propulsion (PEEP) Lab seeks to create opportunities for undergraduate engineering students across the United States to learn about and participate in developing new electric propulsion technologies. PEEP students have designed and built two Hall effect thrusters [2, 3], the latter of which contained a novel 3D printed propellant diffuser. In this paper, the team reports the initial results of testing this diffuser and outlines next steps for improving that integral component.

*Engineering Physics, class of 2023
†Electrical Engineering, class of 2025
‡Mechanical Engineering, class of 2026
§Mechanical Engineering, class of 2025
¶Mechanical Engineering, class of 2024
‖Professor of Mechanical Engineering
**Physics, class of 2024
††Business and Entrepreneurship, class of 2025

II. Hall Thruster Plasma Formation

In a Hall effect thruster, neutral propellant atoms are ionized via electron bombardment. This is achieved in the following manner:

1) The Hall thruster uses an axial electric field to draw electrons into the thruster channel. These electrons are often emitted by an external plasma source called a cathode.
2) A radial magnetic field traps the electrons as they enter the thruster channel; electrons experience a Lorentz force-enabled $E \times B$ drift which causes the electrons to move in a circular path. This azimuthal flow is the Hall current that gives the Hall thruster its name.
3) Neutral gas atoms injected from the bottom of the channel collide with this swirling cloud of high-energy electrons and become ionized (electron bombardment), creating the plasma.

The classical model for electron bombardment describes electrons which collide with propellant atoms, lose kinetic energy, slow down, and fall towards the anode, regaining kinetic energy as they fall through the electric potential (until they finally strike the anode). However, this classical description is incomplete as many more electrons fall through the magnetic field than would be expected in so-called "anomalous electron transport" [4], which is an open field of study.

Only ionized particles are accelerated by the thruster's electric field (and meaningfully contribute to the thrust), hence optimizing the number of propellant atoms (neutrals) ionized increases the efficiency of the whole thruster. The ionizing interactions between the neutrals and the trapped electrons are captured by the ionizing mean free path of the neutrals. This mean free path is the average distance a neutral atom travels before experiencing an ionizing collision with an electron and may be found by the following equation, provided by Goebel and Katz [5]:

$$\lambda = \frac{v_n}{n_e \langle \sigma_i v_e \rangle} \tag{1}$$

where v_n is the axial velocity of a neutral propellant atom, n_e is the electron number density, and $\langle \sigma_i v_e \rangle$ is the experimentally-determined ionization cross section of the propellant, which quantifies the probability of ionization for a collision with an electron at a particular electron temperature. Goebel and Katz [5] further report that the relationship between plasma length and fraction of ionized neutrals is

$$L = -\lambda \ln(1 - p) \tag{2}$$

where p is the decimal fraction of neutrals that leave the thruster ionized. Rearranging Equation 2 yields the relationship

$$p = 1 - e^{-L/\lambda} \tag{3}$$

Therefore, for a thruster of a given plasma length, L, the percentage of propellant ionized is maximized by reducing the mean free path, λ. A mean free path variable that is easy to control is the axial velocity of the neutrals, v_n. Referring to Equation 1, when v_n is reduced, λ is reduced, and p is therefore increased.

An alternate way to conceptualize this relationship is that reducing the axial velocity increases the longevity, or lifetime, of the propellant within the channel. The longer a neutral lasts inside the channel, the higher the chance that it collides with a high-energy electron before escaping the thruster.

III. Diffuser Design

Given that injecting the propellant into the thruster channel with minimum axial velocity will maximize propellant ionization, a propellant diffuser can be designed accordingly: the team designed a diffuser that injects propellant azimuthally, instead of directly axially. Recent experiments from researchers at the Harbin Institute of Technology in China have confirmed that azimuthal injection reduces propellant axial velocity [6] and increases both the ionization and anode efficiencies of krypton propellant [6, 7].

Azimuthal injection was achieved by pushing propellant through two inlet holes into a toroidal plenum (to more evenly distribute the inlet pressure). Branching off of this plenum were four helical channels that guided the propellant into an azimuthal trajectory. A computer aided design (CAD) model of this diffuser geometry is shown in Fig. 1. The diffuser used in this experiment contained only four relatively large tubes because the team originally intended to manufacture the diffuser from a ceramic material that required particular sizes of interior features and wall thicknesses.

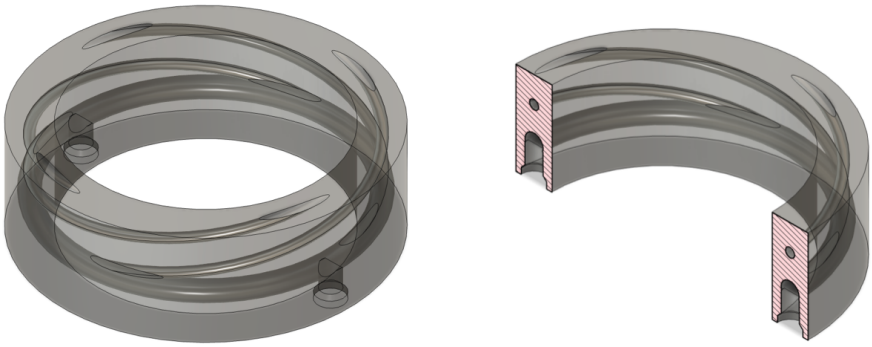

Fig. 1 CAD model and section view of the diffuser used during testing. This figure by Braden Oh, et al. [3] is licensed under CC BY 4.0.

The design for this diffuser necessitates additive manufacturing. The interior features that connect the toroidal plenum to the outlet ports are angled and curved in nature, making them impossible to manufacture with conventional machining techniques. This constraint made additive manufacturing necessary to achieve the design outlined in CAD. Three versions of the depicted diffuser were successfully prototyped: one from a low temperature stereolithographic (SLA) resin; one from a high temperature SLA resin called Somos PerFORM; and one from direct metal laser sintered (DMLS) AlSi10Mg aluminum alloy. The DMLS prototype was black anodized after printing to ensure the surface was electrically insulative. Images of these prototypes are shown in Fig. 2. A fourth prototype was attempted to be made from a Formlabs SLA resin with suspended ceramic particles (the part would be sintered post-printing to fuse the ceramic), but the resin repeatedly clogged the Formlabs printer's dispenser valves and prevented printing.

Fig. 2 The three types of prototyped diffusers. From left to right: low temperature SLA resin, Somos PerFORM, and black anodized aluminum. This figure by Braden Oh, et al. [3] is licensed under CC BY 4.0.

The diffuser made from PerFORM was of greatest interest to the team for two reasons. First, PerFORM is an inexpensive material that can be used for very rapid prototyping. Second, PerFORM has previous space propulsion heritage; PerFORM has previously been used to manufacture cold gas thrusters for CubeSats [8] (demonstrating an ability to run high-pressure propellant under vacuum), and to hold small Hall thrusters during ground testing [9] (demonstrating an ability to survive high temperatures under vacuum). Because it had not been used to directly manufacture a component within an electric propulsion engine, this experiment would serve as an initial demonstration.

PerFORM has a deflection temperature under load (DTUL) of 268°C. Flame and/or burning temperature data for PerFORM is not publicly available. Although 268°C is lower than temperatures typically experienced inside a Hall thruster, the design of the thruster used for testing placed the diffuser under no mechanical load, so it was expected that the true failure temperature of the part would be significantly higher.

IV. Live-Fire Testing

A PerFORM diffuser was integrated into a 50mm Hall thruster built by a team at Olin College in 2022[3]. The diffuser was seated at the bottom of the main boron nitride channel with its top surface directly contacting the steel anode. A thermocouple was inserted through a hole in the channel wall to contact the side of the diffuser. Two brass pipes fed krypton gas into the rear of the diffuser. An image of the assembled thruster is shown in Fig. 3.

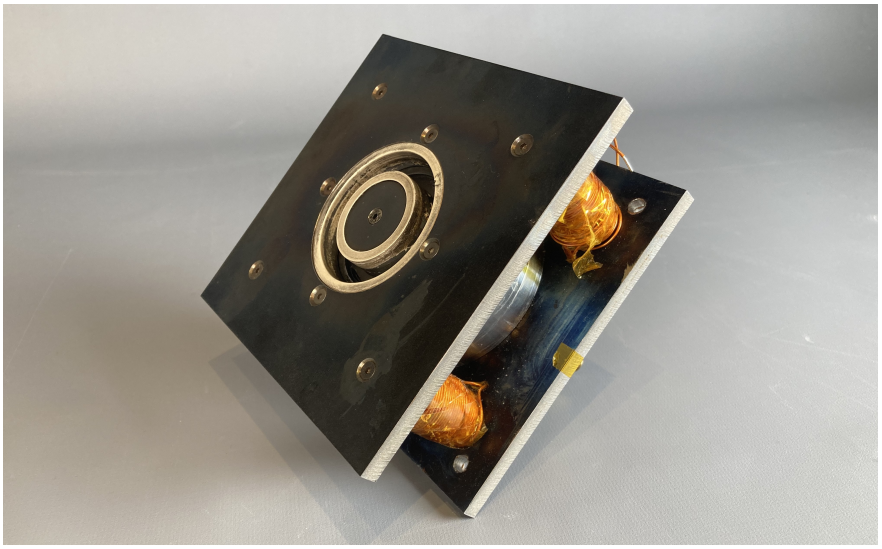

Fig. 3 A post-firing photograph of the thruster used to test the diffuser.

This thruster system was tested under live-fire conditions. During these tests the diffuser dispensed 10-20 SCCM of propellant at thruster power levels in the range of 100-200 W. Early on in the testing process an anomalous high voltage on the diffuser thermocouple lines caused the thermocouple amplifier chip to burn out. Out of concern for the safety of the operators, the thermocouple lines were disconnected and isolated and direct temperature data could not be taken.

The operational plasma showed distinct regions of brightness corresponding to the exit apertures of the diffuser, but the color became more uniform further away from the surface of the anode, as shown in Fig. 4. This gradient indicates highly unequal propellant distribution by the present design; nevertheless, the diffusion was uniform enough for a steady thrust plume to be generated and sustained.

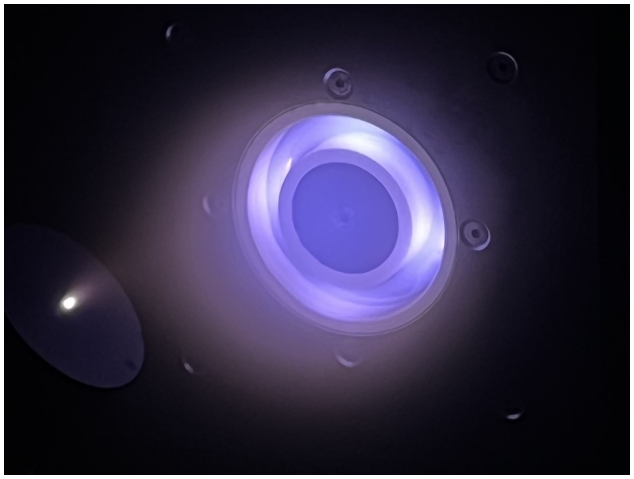

Fig. 4 Photograph of a krypton plasma formed during testing. Bright spots correspond to regions of high plasma density.

An additional phenomenon observed while testing the thruster was a rotating spoke instability[4] with a frequency of approximately 1 Hz that manifested at low magnetic field strengths. Four still frames extracted from video of this instability are shown in Fig. 5. The spoke instability's ability to travel all the way around the circumference of the channel is further evidence of the diffuser's successful operation.

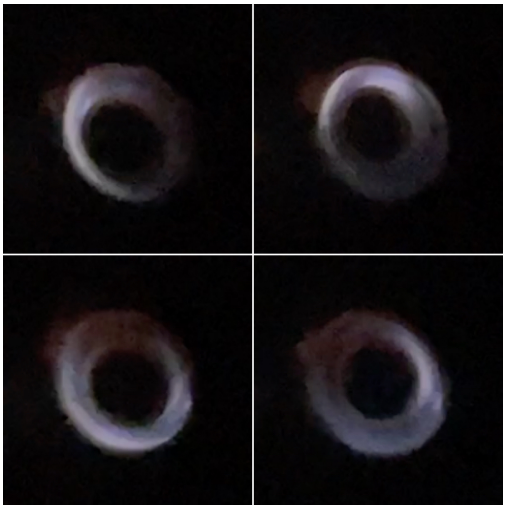

Fig. 5 Still frames from video of a clockwise-rotating spoke instability.

Following testing, the thruster was disassembled so that the diffuser could be inspected. As shown in Fig. 6, charring and fracturing was visible on one half of the diffuser while the other half was distinctly clean. It is not readily apparent whether the fractures, which are presumed to have been created by thermal stress, penetrated deeply enough to penetrate the diffusion pathways.

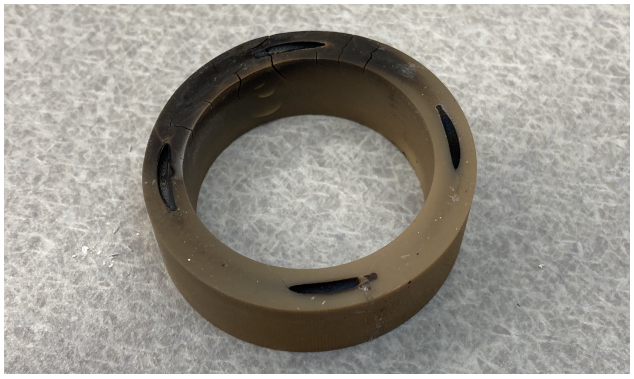

Fig. 6 Photograph of the diffuser post-firing.

The presence of charring and fracturing on only one side of the diffuser implies that thermal energy was either excessively accepted or unable to be rejected from that side alone. It is notable that the side of the diffuser that experienced charring was directly opposite the side which accommodated a wire used to close the anode electrical connection, as shown in Fig. 7.

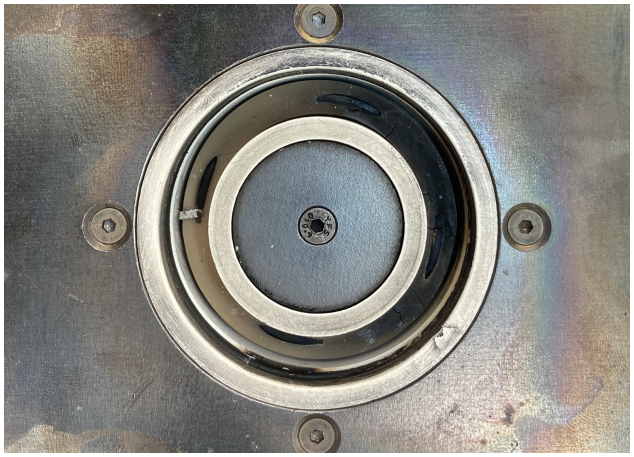

Fig. 7 Photograph of the thruster with the anode plate removed. Charring and thermal damage (right) is exactly opposite the side of the diffuser which accommodates a wire (left) used to close the electrical connection to the anode.

A possible explanation for this orientation of thermal damage is that the electrical wire pushed the diffuser off of its axial alignment and into the channel wall. The channel walls are made from thermally conductive boron nitride and contact the thruster plasma, and thus reach very high temperatures. As the electrical wire knocked the diffuser off-axis, it simultaneously pressed the charred side into the exterior channel wall and served as a thermal standoff between the 'clean' side of the diffuser and the channel wall. This excessive thermal conduction could have led to the failure observed. Additional experiments will be conducted to verify this hypothesis and search for a diffuser design that is robust to this failure mode.

V. Next Steps

This experiment demonstrates that although PerFORM can be used to 3D print a critical gas diffusion component inside of a Hall thruster, as a material it is susceptible to charring and thermal fracturing. To further study its applicability to Hall thruster components, three future experiments are being planned:

1) Validate the electrical wire hypothesis by live-fire testing a diffuser with modified geometry.
2) Identify the charring and thermal fracturing temperatures of PerFORM by baking blocks in a high-temp oven.
3) Determine the usable lifetime of PerFORM as a Hall thruster channel.

Experiment one will be to validate the electrical wire hypothesis for the charring observed. This will be achieved by manufacturing a diffuser with a narrower radial width than the one previously tested. The new diffuser will also have thin ribs on the outside surface to prevent high-surface-contact thermal conduction from occurring.

Experiment two will be to identify the exact failure temperature of PerFORM. This will be found by baking small 3D printed blocks in a furnace at increasingly high temperatures until charring and thermal fracturing is observed.

Experiment three will be to determine how long PerFORM can be used within a Hall thruster at high temperature. This will be achieved by 3D printing an entire thruster channel from PerFORM and observing how the material breaks down after various lengths of live fire time. The purpose of this test is to find out whether PerFORM can feasibly be used to prototype new designs for thruster components in a cheap and rapid way, prior to manufacturing out of a more delicate or expensive material.

VI. Conclusion

This initial experiment demonstrates that an azimuthal gas diffuser 3D printed out of PerFORM is indeed capable of delivering sufficient propellant flow to run a 100-200W Hall thruster at steady state for brief periods of time. However, PerFORM is susceptible to burning and cracking at very high temperatures, indicating that careful thermal management must be considered when using PerFORM to directly manufacture Hall thruster components. Future experiments are being planned to identify the exact temperatures and lengths of time for which components manufactured from PerFORM may be used within a Hall thruster without charring or fracturing.

Acknowledgments

The authors thank Prof. John Williams, Seth Thompson, and their team at Plasma Controls LLC. for donating the cathode and time that were so critical to igniting the Hall thruster. The authors thank Dr. Dan Goebel for his invaluable insights throughout the development of the thruster and Mahderekal Regassa for her participation in designing and manufacturing the prototype diffusers used in this study.

This work was supported by the Massachusetts Space Grant, Draper Labs, and a Babson Olin Wellesley (BOW) Presidential Innovation Grant.

References

[1] Choueiri, E. Y., "New dawn for electric rockets," *Scientific American*, Vol. 300, No. 2, 2009, pp. 58–65.

[2] Oh, B. K., Kunimune, J. H., Spicher, J., Anfenson, L., and Christianson, R., "Undergraduate Demonstration of a Hall Effect Thruster: Self-Directed Learning in an Advanced Project Context," *2020 ASEE Virtual Annual Conference Content Access*, 2020.

[3] Oh, B., Countryman, A., Regassa, M., Clowes, A., Miner, G., Kemp, S., McAneney, S. Klein, M., and Lee, C., "Design, fabrication, and testing of an undergraduate hall effect thruster," *Journal of Electric Propulsion*, Vol. 2, No. 1, 2023, p. 6.

[4] McDonald, M. S., "Electron Transport in Hall Thrusters," Ph.D. thesis, University of Michigan, Ann Arbor, MI, 2012.

[5] Goebel, D. M., and Katz, I., *Fundamentals of electric propulsion: ion and Hall thrusters*, John Wiley & Sons, 2008.

[6] Xia, G., Li, H., Zhu, X., Ning, Z., Chen, S., Yu, D., and Zhou, C., "Effects of rotating supply mode on the ionization parameters of a krypton Hall thruster," *Vacuum*, Vol. 181, 2020, p. 109664.

[7] Xia, G., Li, H., Ding, Y., Wei, L., Chen, S., and Yu, D., "Performance optimization of a krypton Hall thruster with a rotating propellant supply," *Acta Astronautica*, Vol. 171, 2020, pp. 290–299.

[8] Hart, S. T., Daniel, N. L., Hartigan, M. C., and Lightsey, E. G., "Design of the 3-D Printed Cold Gas Propulsion Systems for the VISORS Mission," *2022 AAS GNC Conference, Breckenridge, CO, USA*, 2022.

[9] Bretti, M. A., "Progress and Developments of Ultra-Compact 10 Watt Class Adamantane Fueled Hall Thrusters for Picosatellites," *Proceedings of the 37th International Electric Propulsion Conference. Cambridge. IEPC-2022-349*, 2022.

]

Part IV
Protecting Geostationary Satellite Services using the Equivalent Power Flux Density (EPFD) Algorithm

Protecting Geostationary Satellite Services using the Equivalent Power Flux Density (EPFD) Algorithm

Audrey Lee, Charlie Babe, Michael Remley, Andrew Phillips, Kate McCurley, Ben Eisenbraun, Kaitlyn Fleming, Kristtiya Guerra, Carlota Ramiro de Huelbes, * Whitney Lohmeyer †
Olin College of Engineering, Needham, MA 02492

Article 22 of the International Telecommunications Union (ITU)'s Radio Regulations defines Equivalent Power Flux Density (EPFD) limits in order to ensure protection from harmful interference from non-geostationary (NGSO) systems into geostationary (GSO) satellites and earth stations (ES). This paper focuses on the ITU's EPFDdown algorithm and its applications in interference mitigation. EPFDdown is defined as the EPFD emitted from the NGSO satellite that is downlinked into the victim GSO ES. The ITU's Radiocommunication Sector (ITU-R) has published Recommendation S.1503 to provide guidance for satellite operators developing software to evaluate EPFD limits. The Olin Satellite + Spectrum Technology & Policy (OSSTP) Group has worked to implement the EPFDdown algorithm in ITU-R S.1503 to analyze EPFDdown and validate the submissions of three NGSO systems: ARISTARCHUS, METHERA-A, and 3ECOM-1. Using the EPFD validation process, one can ensure that incumbent and planned GSO networks can operate without harmful interference from NGSO systems, reducing data loss. This paper evaluates each step of EPFD analysis in order to validate different systems' EPFDdown showings and aims to make complex EPFDdown validation more accessible to incumbent and planned satellite systems.

I. Introduction

Equivalent power flux density (EPFD) is a metric to ensure that geostationary (GSO) satellites and earth stations (ES) are protected from harmful interference from non-geostationary (NGSO) systems [1]. EPFD is a function of the power flux density (PFD) amplified or attenuated by the antenna gains of the NGSO and victim GSO systems. Article 22 of the International Telecommunications Union's (ITU) Radio Regulations (RR) defines such interference limits for NGSO networks in down, up, and inter-satellite directions. The ITU's Recommendation ITU-R S.1503-3 details a method to perform interference calculations for all three directions and confirm conformance to Article 22 limits.

EPFDdown quantifies the potential interference from the NGSO's satellite transmitter into the GSO ES receiver. As an NGSO satellite transmits towards its NGSO ES, some of the transmitted power inevitably hits the GSO ES and causes interference that impacts the availability of the GSO network [2]. Two types of EPFD limits exist: aggregate and operational. Aggregate limits account for interference from multiple NGSO systems, whereas operational limits account for the EPFD produced from a single system. This paper focuses on operational EPFDdown validation limits, which are statistical in nature and are defined as a given power level that can be exceeded up to a certain percentage of time. Most EPFDdown operational limits are absolute and never to be exceeded in order to protect existing GSO satellite systems orbiting Earth.

This paper also introduces an accessible translation of ITU-R S.1503 and shows how the EPFDdown validation algorithm is implemented. EPFDdown results from the Olin Satellite + Spectrum Technology & Policy (OSSTP) Group's MATLAB implementation of ITU-R S.1503 are compared to the results from both types of ITU validation software: Transfinite and Agenium. Both are companies that developed their own EPFD validation software for use by the ITU. Because both companies developed their algorithms independently, the two algorithms result in slightly different EPFD calculations. [3].

II. High Level Overview of EPFDdown Calculation Process

When calculating EPFDdown for a satellite system, the user inputs the NGSO, GSO, and Article 22 parameters as seen in Figure 1. This algorithm specifically focuses on the interference between one NGSO system and one GSO

*Research Assistants, Olin Satellite + Spectrum Technology & Policy Group, 1000 Olin Way, Needham, MA 02492
†Lab Director, Olin Satellite + Spectrum Technology & Policy Group, 1000 Olin Way, Needham, MA 02492

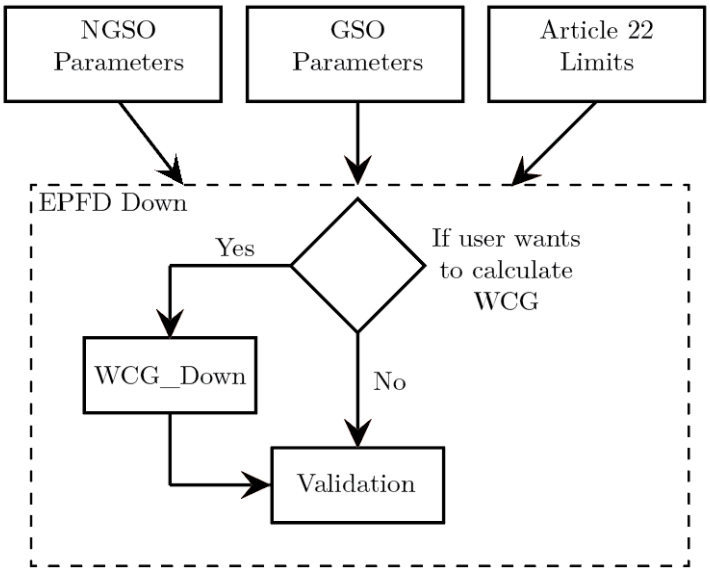

Fig. 1 Block diagram of the EPFDdown Calculation Process

system [4]. The user has the choice to select either the Worst Case Geometry (WCG) Algorithm to find the worst possible interference scenario for the GSO ES or provide geometric related parameters manually. These positions are then passed into the main EPFDdown validation algorithm, which then validates whether or not the system is within the defined ITU Article 22 limits.

1. Initializing Parameters

The first step of the EPFDdown calculation process is to input the necessary NGSO, GSO, and Article 22 parameters. These parameters are pulled from two Access files, Mask.mdb, which contains the satellite power flux density (PFD) mask, and SRS.mdb, which contains all other parameters. The PFD mask defines the power radiated from an NGSO system. SRS refers to the ITU Space Radiocommunications Service (SRS) database. The NGSO System parameters contain the NGSO system's orbital characteristics, shown in Table 2 along with the other required NGSO parameters, including the PFD mask. The Parameter Value column shows the expected value of the parameter, either a variable or a Yes/No statement. Table Name in File lists which table the parameter is under in the Access file. Parameter Name in File refers to the name of the parameter in the Access file.

GSO parameters comprise the next set of inputs, which are summarized in Table 1 along with their method of input.

Table 1 GSO System Parameters

GSO Parameter	Parameter Value	Method of Input
GSO ES gain pattern	GSO_ES_PATTERN	User
Frequency (MHz)	F_DOWN	User
GSO ES dish size (m)	GSO_ES_D_ANT	User
Reference bandwidth (kHz)	REFBW	User
Array of EPFDdown values (dB(W/m^2*BW_ref))	epfd_down	Corresponding Article 22 Limits
Array of EPFDdown percentages (%)	perc_time_epfd_down	Corresponding Article 22 Limits
GSO satellite longitude (deg)	GSO_SAT_LONG	Calculated
GSO ES latitude (deg)	GSO_ES_LAT	Calculated
GSO ES longitude (deg)	GSO_ES_LONG	Calculated

Users are then able to select GSO parameters and the type of runs to validate via a graphical user interface as shown for the Agenium software in Figure 2. These parameters include:

- GSO ES gain pattern: GSO_ES_PATTERN
- Frequency: F_DOWN (MHz)
- GSO ES dish size: GSO_ES_D_ANT (m)
- Reference bandwidth: REFBW (kHz)

The user selects the desired EPFDdown run and on which parameters the algorithm should be based, including the GSO ES gain pattern, GSO_ES_PATTERN, which will be either fixed-satellite service (FSS) or broadcasting-satellite service (BSS). Referring back to Table 1, an example for the frequency, F_DOWN (MHz), is 17800.02 MHz, an example of a GSO ES dish size, GSO_ES_D_ANT (m), is 1.00 m and an example of reference bandwidth, REFBW (kz), is 1000 kHz.

The corresponding Article 22 limits will be used based on the run(s) the user selects [5]. These include the output variables:

- Array of EPFDdown values: epfd_down ($dB\frac{W}{m^2(BW_{ref})}$)
- Array of EPFDdown percentages: perc_time_epfd_down (%)

which are defined in Table 1.

The user then selects the frequency, radiation pattern, antenna diameter, and reference bandwidth for their run. These inputs will correspond to the "EPFD" and "Percentage of time during which EPFD may not be exceeded" columns in Article 22's database. For example, if the user were to pick a GSO ES PATTERN of ITU-R S.1428-1 [6], meaning FSS, a F_DOWN of 10.7 GHz, a GSO_ES_D_ANT of 60 cm, and a reference bandwidth (REFBW) of 40 kHz, then according to Article 22 [5], the corresponding EPFDdown values and EPFDdown percentages chosen would be the first row of the two corresponding columns in Table 22-1A. By counting how many times each EPFD level occurs, a probability density function (PDF) and cumulative distribution function (CDF) can be created and compared against the EPFD thresholds. The PDF turns each bin from a count to a percentage by dividing its count by the total number of observed EPFD levels, so that the PDF is the probability that a given EPFD level occurs. The sum of all EPFD levels in the PDF should be 100%. The CDF sums the PDF from the highest EPFD level to a given level; this sum determines the probability that an EPFD level is exceeded.

There are also GSO parameters that are calculated to obtain the worst possible location of the GSO satellite and ES for the systems:

- GSO satellite longitude: GSO_LONG (deg)
- GSO ES latitude: GSO_ES_LAT (deg)
- GSO ES longitude: GSO_ES_LONG (deg)

The next step is to calculate the WCG, which is the GSO satellite longitude and ES latitude and longitude at which the maximum level of EPFD occurs.

Table 2 NGSO System's Orbital Characteristics

NGSO Parameter	Parameter Value	Table Name in File	Parameter Name in File	Extra Info
Satellite PFD mask	N/A	Masks	F_mask (where f_mask = P for PFD masks)	See part C in 1503-2 for more info
Transmit center frequency (GHz)	F_DOWN_sat	mask_info	Freq_min, freq_max	For the middle frequency: (freq_min + freq_max)/2
Exclusion zone parameter	Alpha or X	non-geo	f_x_zone	If f_x zone = Y, then x_zone = Alpha, else x_zone = X
Exclusion zone angle (deg)	MIN_EXCLUDE	non-geo	x_zone	See exclusion zone parameter for what the x_zone value represents
Max number of satellites operating at f_sat by latitude	N_co[lat]	sat_oper	nbr_op_sat	
Number of NGSO satellites	N_sat	orbit	Nbr_sat_pl	Sum all nbr_sat_pl in tcolumn for all N satellites
Orbit has repeating ground track maintained by station keeping	Yes or No	orbit	f_stn_keep	
Admin. supplying specific node procession rate	Yes or No	orbit	f_precess	
Station keeping range for ascending node as half total range (deg)	W_delta	orbit	keep_rnge	
Phase angle (deg)	phase_ang	phase	phase_ang	
Minimum operating height (km)	H_MIN	grp	elev_min	

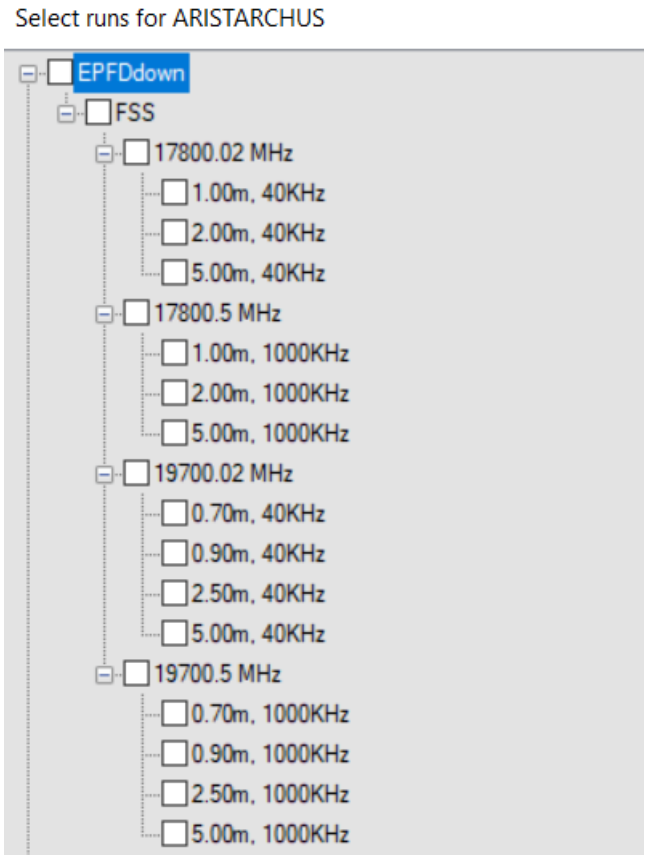

Fig. 2 Example AGENIUM User Interface for the user to select validation runs

III. EPFDdown Validation Methodology

The EPFDdown algorithm requires the user to input an ITU SRS Database containing the NGSO system parameters, a PFD mask including the PFD values calculated for the NGSO system, and the ITU's Article 22 EPFDdown limits. The satellite PFD mask is defined as the maximum PFD generated by any space station in the interfering NGSO system as seen from any point at the surface of the Earth [7].

The EPFDdown algorithm inputs the parameters shown in Figure 1 such as information from the PFD mask, NGSO system parameters, and GSO system parameters. The algorithm then determines the highest PFD value for a specific geometry: the location of the NGSO satellite in relation to the GSO satellite and ES. This PFD value, along with the gain value of the satellite, is then used to calculate the EPFD. This EPFD value is computed and then counted over a time step, creating a statistic of how often the EPFD value occurs. This statistic is then plotted against the Article 22 limits [5], validating the NGSO's compliance with Article 22 and ensuring the protection of incumbent GSO satellite systems.

A. The EPFDdown Validation Algorithm Methodology

This section provides an overview of the ITU-R S.1503-2 [8] and ITU-R S.1503-3 [7] documentation used to calculate the interference from a system of NGSO satellite transmitters into a GSO ES. ITU Recommendation S.1503-3 is an updated version of the S.1503-2 recommendation. The process outlined in the documentation was implemented in MATLAB, from which the results of the implementation were compared to Agenium and Transfinite's implementation of the algorithm's results. Figure 3 shows a high level block diagram of the EPFDdown Validation Algorithm.

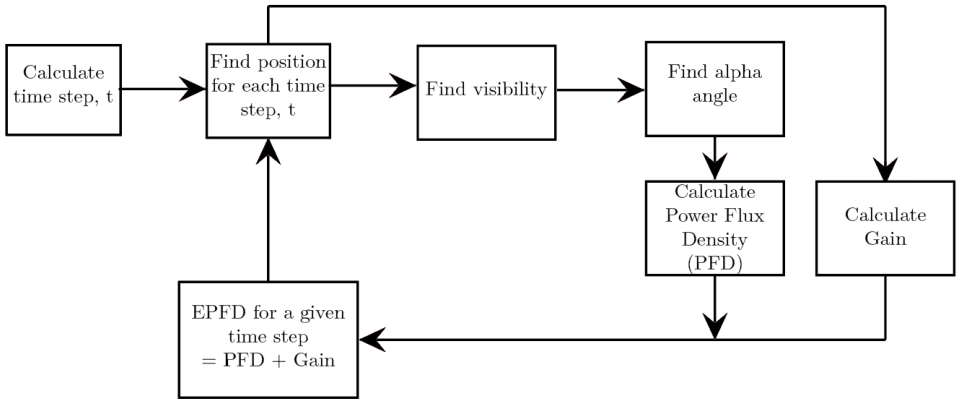

Fig. 3 Block diagram of the EPFDdown Validation Algorithm

The first step in the algorithm is to input the same input parameters described in Section II (e.g. transmit center frequency, exclusion zone parameter, exclusion zone angle). The next step is to create a data structure to store EPFD values for the PDF calculation, which will be used to calculate the CDF. Once the initial data structure is created, the algorithm will then determine the number of time steps needed to complete a run of the simulated satellite system.

1. Determining Time Steps for Simulation

The number of time steps needed to complete a run (T_{run}) and the step size (T_{fine}) are calculated, which are defined in section D.4 in ITU-R S.1503-2 [8]. If the user selects "dual-time step", then the N_{coarse} step size for $T_{coarse} = T_{fine} \cdot N_{coarse}$ is calculated. N_{coarse} is the number of "fine" time steps that can be skipped to obtain a "coarse" estimate of the simulation. If "dual-time step" is not selected, N_{coarse} is set to 1, and $T_{coarse} = T_{fine}$.

After each time step, the current amount of time, t, in the simulation is incremented by T_{step}. If the user selects "dual-time step", T_{step} needs to be calculated for all time steps until the current time, t, equals the end of the simulation T_{run}. Furthermore, $T_{step} = T_{coarse}$ if all of the following conditions are met:

- The user selects "dual-time-step"
- If it is not the first time step (t $\neq$ 0)
- If the $G_{RX}(\phi)$, where $G_{RX}(\phi)$ is the gain with respect to off-axis angle ϕ, for the previous time step is not within 30 dB of the max peak gain (G_{max}) [7]

If any of the aforementioned conditions are not met, then $T_{step} = T_{fine}$.

2. Simulating GSO and NGSO systems

The next step is to simulate the position of the GSO ES relative to the center of the Earth, the position of the GSO satellite, and the positions of the NGSO satellites at time, *t*. Based off of the input parameters from the NGSO and GSO system, the positions and orbits of both the GSO and NGSO satellites as well as the GSO ES position can be calculated, so that the NGSO satellites that are visible to the GSO ES can be identified in the current time step.

Figure 4 illustrates system visibility between an NGSO satellite and a GSO satellite. Each satellite's cone represents its area of coverage on Earth.

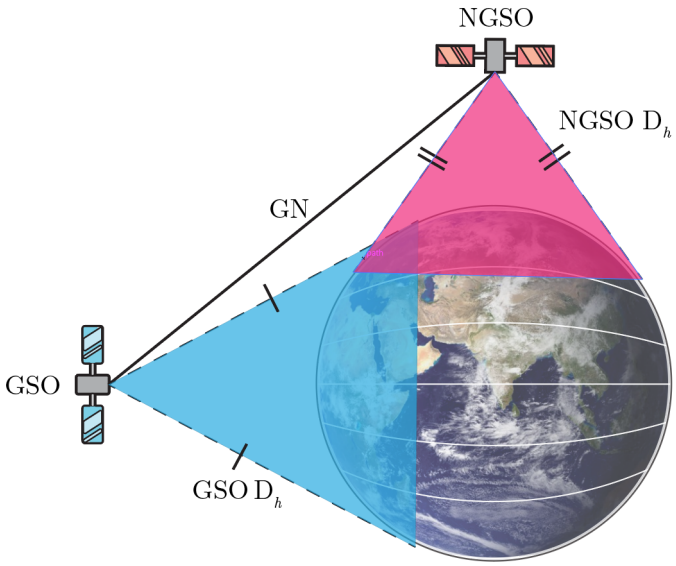

Fig. 4 Diagram of Visible Systems

If the areas of coverage overlap, then the satellites are visible to each other. This is represented mathematically by the direct distance between the two satellites, GN, and the line from the satellite to an edge of its visibility on Earth, D_h. If the sum of the two satellites' D_h values is greater than GN, then the satellites are visible to each other. Figure 5 shows a case where two satellites are not visible to each other.

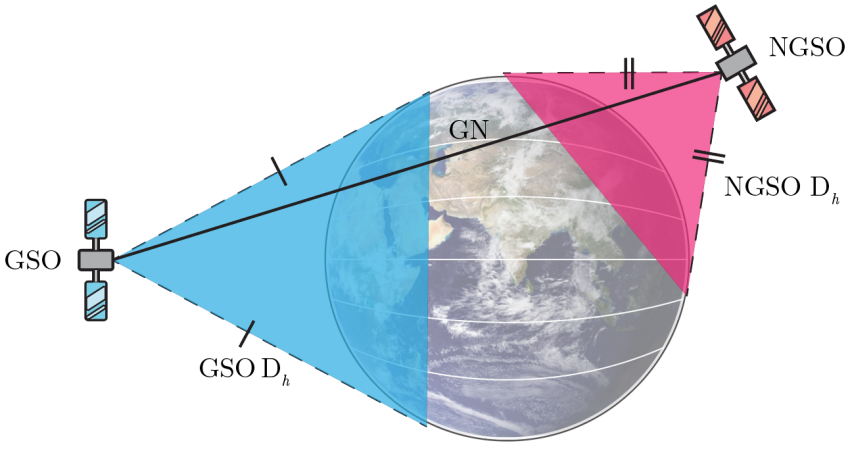

Fig. 5 Diagram of Non-visible Systems

For each NGSO satellite, *i*, that is visible from the GSO ES, the EPFDdown values for the current time step are

calculated and stored.

The algorithm determines the input parameters needed to calculate the PFD from the "type" of PFD mask, either "alpha_deltaLongitude" or "azimuth_elevation", which can be found in the Mask.mdb file. Regardless of the type of PFD mask, the latitude of the NGSO must be calculated using the following steps:

If the type of PFD mask is “azimuth_elevation”, calculate the azimuth and elevation of NGSO i.

If the type of PFD mask is “alpha_deltaLongitude”:

- Calculate the minimum α (or X depending on the variable f_x_zone from Table 2) angle in which there is a point, P, on the GSO arc that minimizes the angle between the NGSO and GSO relative to the ES. More information on calculating the minimum alpha angle will be provided in the next section of this paper.
- Calculate the delta longitude of the NGSO where delta longitude is equal to the longitude of the found minimum alpha angle minus the longitude of the NGSO.

3. Finding Minimum Alpha & the Corresponding Delta Longitude

The alpha angle, α, is the angle between two vectors from a GSO ES, one to an NGSO satellite and one to a test point on the GSO arc, illustrated in Figure 6.

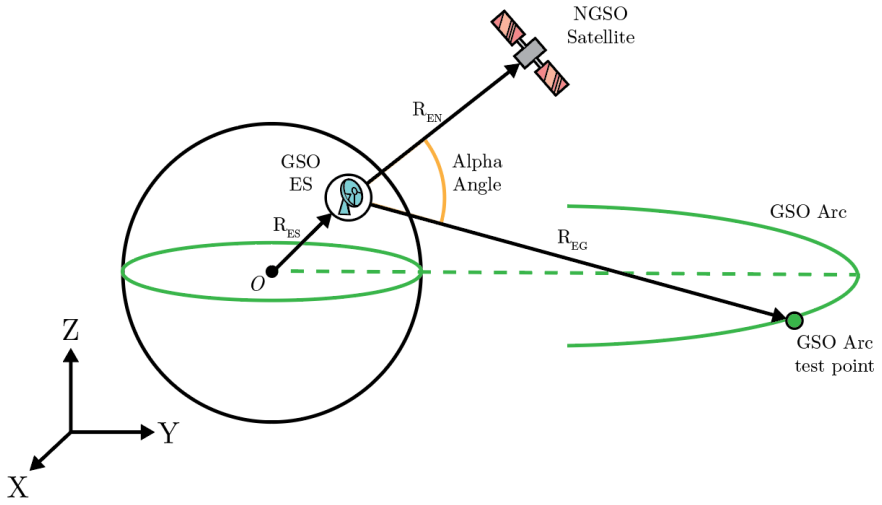

Fig. 6 Diagram of the Alpha Angle

Multiple alpha angles are calculated from testing multiple points on the GSO arc. A smaller alpha angle indicates more interference between a GSO ES and an NGSO satellite is likely, so the minimum α is calculated to find the WCG.

The first step in calculating the minimum α is to determine its sign. This involves calculating a new vector, R_{EN}, to represent the distance between the NGSO and ES position vectors, R_{NS} and R_{ES}:

$$R_{EN} = R_{NS} - R_{ES} \tag{1}$$

Then, construct the line:

$$R = R_{ES} + \lambda R_{EN} \tag{2}$$

This line connects the GSO ES and NGSO satellite, as shown in Figure 7, and will also intersect the XY plane, on which lie the equator and GSO arc, at the point $R_{z=0}$.

To find $R_{z=0}$, next calculate the ratio, $\lambda_{z=0}$, of the ES z-coordinate to the distance between the NGSO and ES in the z-axis:

$$\lambda_{z=0} = -\frac{ES_z}{R_{en_z}} \tag{3}$$

Hence, Equation 4 is used to find the intersection point between R and the XY plane, which will be used to determine the sign of α.

$$R_z = ES + \lambda_z R_{en} \tag{4}$$

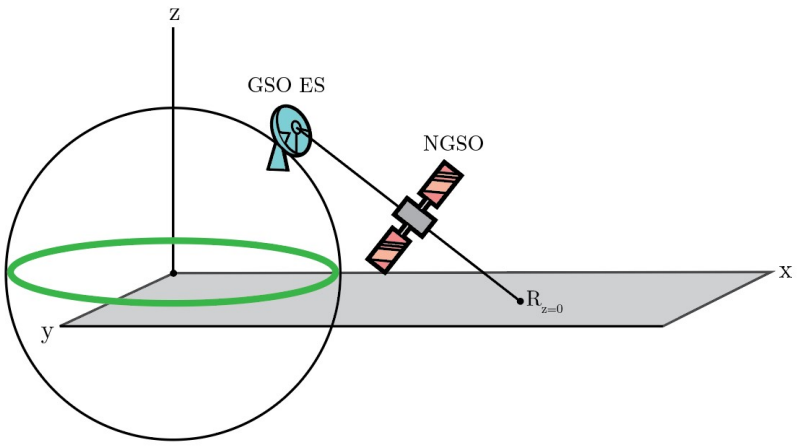

Fig. 7 Diagram of R Intersecting the XY Plane at $R_{z=0}$

Where this point falls with respect to the GSO arc allows us to determine the sign of α. The sign of α also depends on the hemisphere in which the NGSO satellite is located.

If the ES is in the Northern Hemisphere, Figure 8 shows the sign of α based on the location of $R_{z=0}$ relative to the radius of the GSO arc R_{geo}. If $R_{z=0}$ falls within the shaded region, α is greater than 0; if alpha is above the shaded region, α is less than 0. If $R_{z=0}$ falls on the GSO arc, α is 0. It is also necessary to consider the sign of $\lambda_{z=0}$: if $\lambda_{z=0}$ is less than or equal to zero, the sign of α will be negative regardless of the value of $R_{z=0}$.

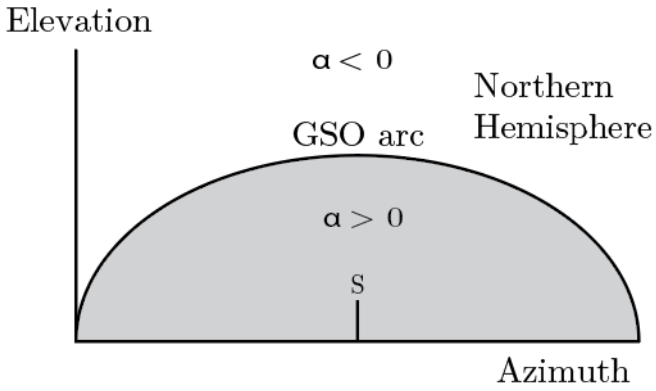

Fig. 8 Sign of α given $R_{z=0}$ with respect to R_{GEO} in the Northern Hemisphere looking South

If the ES is in the Southern Hemisphere, the sign of α is also determined by the signs of $R_{z=0}$ and $\lambda_{z=0}$; however, many of the signs are reversed. Figure 9 shows the sign of α based on $R_{z=0}$. If $R_{z=0}$ falls within the shaded region, α is less than 0; if it's above, α is greater than 0. Again, if $R_{z=0}$ falls along the GSO arc, α is 0. It is once again necessary to consider the value of $\lambda_{z=0}$: if $\lambda z = 0$ is greater than or equal to 0, then alpha will be positive regardless of the value of $R_{z=0}$.

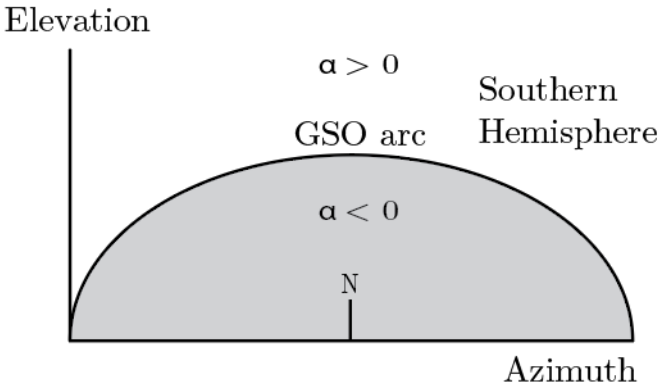

Fig. 9 Sign of α given $R_{z=0}$ with respect to R_{GEO} in the Southern Hemisphere looking North

To find the minimum α, the following mathematical equations are implemented to approximate a range of theta, θ, angles for which the minimum α can be found. Both the θ and delta longitude angles are shown in Figure 10.

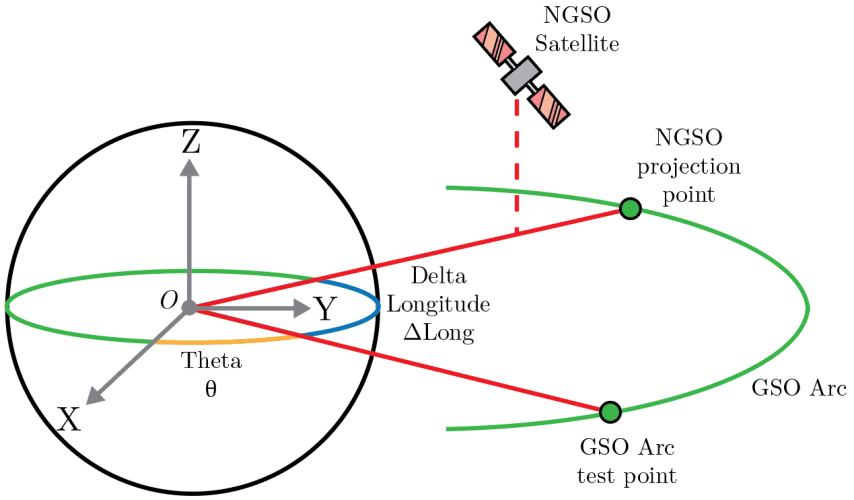

Fig. 10 Diagram of the Theta and Delta Longitude angles

Theta, θ, is the angle from the Earth's X-axis to a vector defined by a test point on the GSO arc. Delta longitude is the angle between the aforementioned vector and a vector defined by two points: the location of the NGSO satellite projected onto the XY plane and Earth's origin. The labeled NGSO projection point is the intersection of this vector with the GSO arc.

The mathematical steps outlined in Equations 5 through 11 walk through the setup for finding the minimum α angle by defining the positions of the GSO ES and NGSO satellite and using them to create the two vectors that define α: one from the GSO ES through a point on the GSO arc and another from the GSO ES through the NGSO satellite. Through the process of defining the relationship between α and θ, the minimum α can be found with respect to θ.

Two position vectors, P and N are defined:

$$P = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} \tag{5}$$

$$N = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \tag{6}$$

where P represents the GSO ES and N represents the NGSO satellite with the center of the Earth as the origin for both systems. Because the ES lies on Earth's surface, its position vector will have a magnitude equal to the radius of the Earth.

Next, G is introduced, the position vector for the GSO satellite on the GSO arc defined by θ:

$$G = \begin{pmatrix} R_{geo} \cos\theta \\ R_{geo} \sin\theta \\ 0 \end{pmatrix} \tag{7}$$

where R_{geo} represents the Euclidean distance from the origin of the Earth to the GSO arc. Note that the z-coordinate is always zero, indicating that the GSO must always lie on the XY-plane.

Using the defined positions of P, N, and G, vectors PN and PG are defined, which connect their respective points.

$$PN = \begin{pmatrix} x_n - x_p \\ y_n - y_p \\ z_n - z_p \end{pmatrix} \tag{8}$$

$$PG = \begin{pmatrix} R_{geo} \cos\theta - x_p \\ R_{geo} \sin\theta - y_p \\ -z_p \end{pmatrix} \tag{9}$$

The relationship between the α and θ angles can then be established by defining the dot product of the two vectors in Euclidean space.

$$\cos\alpha = \frac{PN \cdot PG}{|PN||PG|} \tag{10}$$

α will be minimized for a given θ when:

$$\frac{\mathrm{d}\alpha}{\mathrm{d}\theta} = 0 \tag{11}$$

In Equation 11, theta, θ, gives longitude ranges to minimize alpha, α. The solution must also be better than points on the GSO arc that are barely visible to the ES. Different α values can then be calculated by using all possible θ values to calculate the possible GSO positions. By taking the angles between the NGSO and all possible GSO positions relative to the ES, the minimum α angle can be found.

To obtain the corresponding delta longitude, one must find the corresponding θ that minimizes α and subtract the longitude of the NGSO satellite i from the θ.

4. Calculating PFD and Gain to Calculate EPFD for NGSO Satellite, i

The next step in the algorithm is to calculate the PFD at the GSO ES from the input parameter "PFD mask". A function to calculate PFD is extrapolated from the PFD mask. The function contains the inputs from the "alpha deltaLong" or "azimuth elevation" calculation (depending on the type of PFD mask). One such method to extrapolate a function is to use MATLAB's Scattered Interpolant function with the "nearest" extrapolation method. When passing the α and delta longitude values calculated in Section III.A.3 into this interpolated PFD function, a single PFD value will be returned, which will be used to calculate EPFD.

The off-axis angle, ϕ, between the GSO and NGSO satellite, i, relative to the GSO ES should be calculated to identify the corresponding gain from the gain function. The receive gain, $G_{RX}(\phi)$, and the maximum gain, G_{max}, can be calculated depending on the gain pattern.

The receiver gain, $G_{RX}(\phi)$, is a piece-wise function dependent on the antenna radiation pattern. If the GSO is a fixed-satellite service (FSS) system, different case statements presented in Rec. ITU-R S. 1428 must be used [6], whereas if the GSO is a broadcasting-satellite service (BSS) system, Rec. ITU-R BO.1443 must be used [9]. Figure 11 shows that the gain pattern function chosen is also dependent on the frequency used and the diameter of the GSO ES Antenna. With a specific ϕ value, the corresponding gain value can be obtained from the chosen gain pattern function.

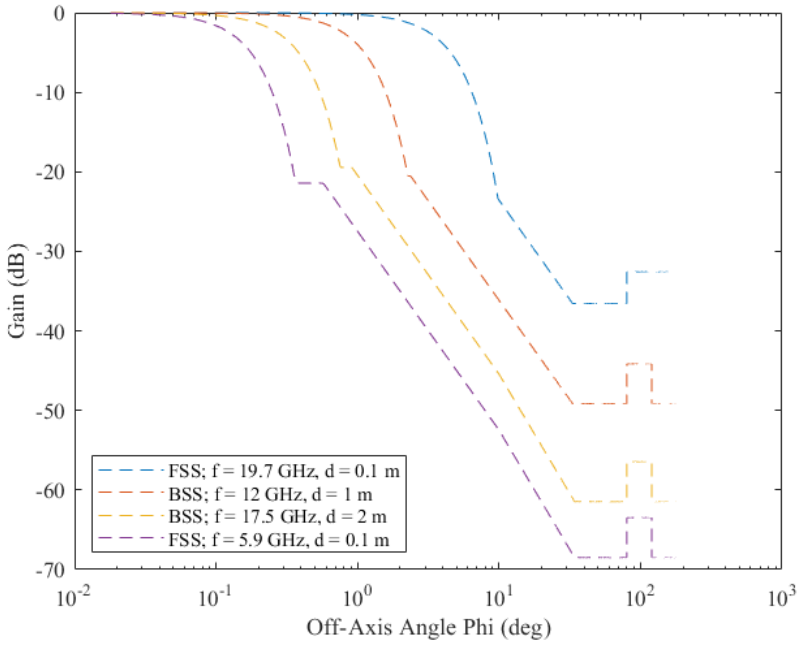

Fig. 11 Examples of Gain Patterns as a function of frequency, GSO ES Antenna Diameter, and service class: fixed-satellite service (FSS) or broadcasting-satellite service (BSS)

The maximum gain, G_{max}, is defined in the ITU Recommendation documents and is used to calculate EPFD.

The EPFD for the current NGSO satellite, i, can be calculated from the PFD, the gain value from the calculated ϕ, and G_{max}

$$EPFD_i = PFD + G_{RX}(\phi) - G_{max} \tag{12}$$

The EPFD value for the *ith* NGSO satellite can be stored in one of two arrays depending on the following conditions:

- If the calculated absolute value of the minimum α angle is outside the exclusion zone (NGSO system params.MIN EXCLUDE) and if the elevation angle of the NGSO relative to the center of the Earth is greater than the input parameter minimum elevation angle (NGSO system params.elev min), store the $EPFD_i$ value in the first array ($EPFD_{i_1}$).
- If the calculated absolute value of the minimum α angle is inside the exclusion zone, one is to store the $EPFD_i$ value in the second array ($EPFD_{i_2}$).

5. Calculating the EPFDdown Value for Time Step t

Once all EPFD values are calculated for all visible NGSO satellites for that time step, the EPFD values of the first array are sorted in ascending order. Then, the total EPFD for that time step is calculated. First, n highest $EPFD_{i_1}$ contributions are summed from the sorted first array where n is the number of satellites operating at the same frequency (NGSO system params. N_{co}), as shown in Equation 13:

$$EPFD_1 = \sum_{i_1=0}^{n} 10^{\frac{EPFD_{i_1}}{10}} \tag{13}$$

All $EPFD_{i_2}$ contributions are separately summed from the second array

$$EPFD_2 = \sum_{i_2=0}^{s} 10^{\frac{EPFD_{i_2}}{10}} \tag{14}$$

where s is the length of the second array.

The total EPFD for that time step is the sum of the two EPFD arrays

$$EPFD = EPFD_1 + EPFD_2 \tag{15}$$

The EPFD calculated is not in decibels, so one must convert the EPFD value back to decibels using

$$EPFD = 10\log(EPFD) \tag{16}$$

The statistics of the PDF are then updated by tracking the number of times the calculated EPFD value occurs. All EPFD values must be rounded down by the nearest 0.1 dB. For every new EPFD value, the EPFD value is recorded with a count of "1". Each time the EPFD value calculated is the same as a previous one, the count for that EPFD value will increment by one.

This whole process is repeated until all time steps are finished (i.e. when $t = T_{run}$).

When all time steps are finished, the program should return the following data:

- Overall "Pass" if the maximum EPFD value does not exceed the maximum Article 22 limit
- Individual Pass/Fail for each EPFD value
- The PDF
- The CDF (which is calculated from the PDF)

If the OSSTP EPFDdown algorithm's run receives an overall "Pass" for the Article 22 limits and the individual EPFD values "Pass", then the algorithm suggests that the satellite services of the GSO systems will be protected from harmful interference from the NGSO system.

B. Validation of Different Satellite Systems

Three NGSO satellite systems: ARISTARCHUS, METHERA-A, and 3ECOM-1 were input into the EPFDdown algorithm as well as the pre-existing EPFDdown programs, Transfinite and Agenium. Transfinite and Agenium are based on an interpretation of the methodology outlined in ITU-R S.1503-2 to validate a system's EPFD compliance [3]. In comparison, the OSSTP Group's algorithm is an interpretation of both the ITU-R S.1503-2 and ITU-R S.1503-3 documentation. All of these satellite systems were already validated as "Pass" by the ITU, so the EPFDdown results from the paper's algorithm and the pre-existing programs' algorithm must also return "Pass."

C. System Validation Results

Figures 12, 13, and 14 show the output of ARISTARCHUS, METHERA-A, and 3ECOM-1's EPFDdown values respectively, which were computed using provided SRS.mdb and Mask.mdb database files for each of the systems. The black line is from the OSSTP Group's EPFDdown algorithm outlined in Section III. The cyan line is constructed using the same inputs as the OSSTP EPFDdown algorithm, but run through Transfinite and Agenium's EPFDdown algorithms. Both algorithms produced the same results for these three satellites, so their results are graphed as a single line. The red dashed line is the Article 22 limits.

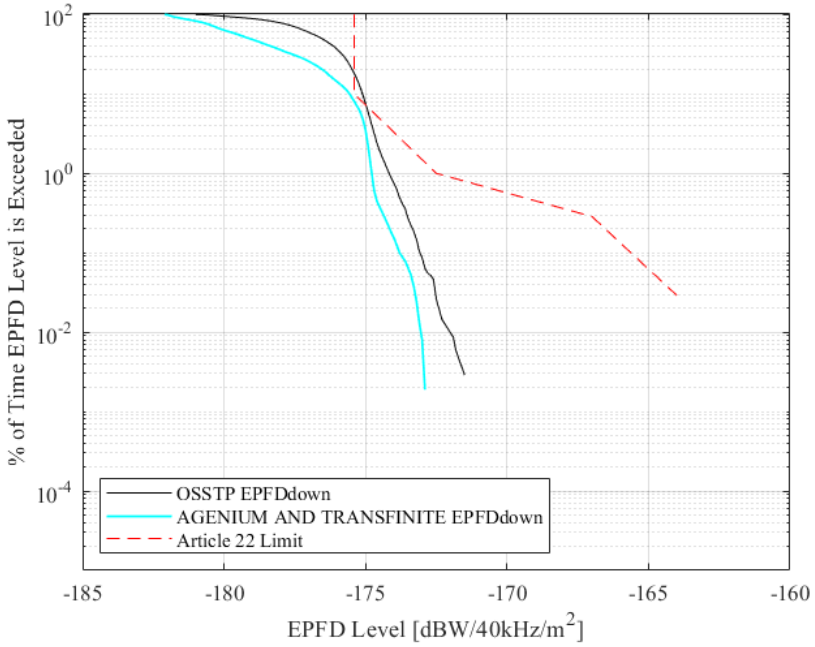

Fig. 12 Validation of ARISTARCHUS's EPFDdown Compliance. This run used a satellite frequency of 17.8 GHz, a dish size of 1m, FSS, and a reference bandwidth of 40kHz.

In Figure 12, both the OSSTP and Transfinite/Agenium results pass the Article 22 limits, and the ARISTARCHUS system is validated. Though the OSSTP EPFDdown curve and the limit visually look like they intersect, the visual intersection was the result of connecting two data points that were both within the the Article 22 limit. In Figure 13, the METHERA-A constellation is examined, which contains 32 satellites in each of its 4 planes. The results show a closer alignment between the OSSTP and Transfinite/Agenium results:

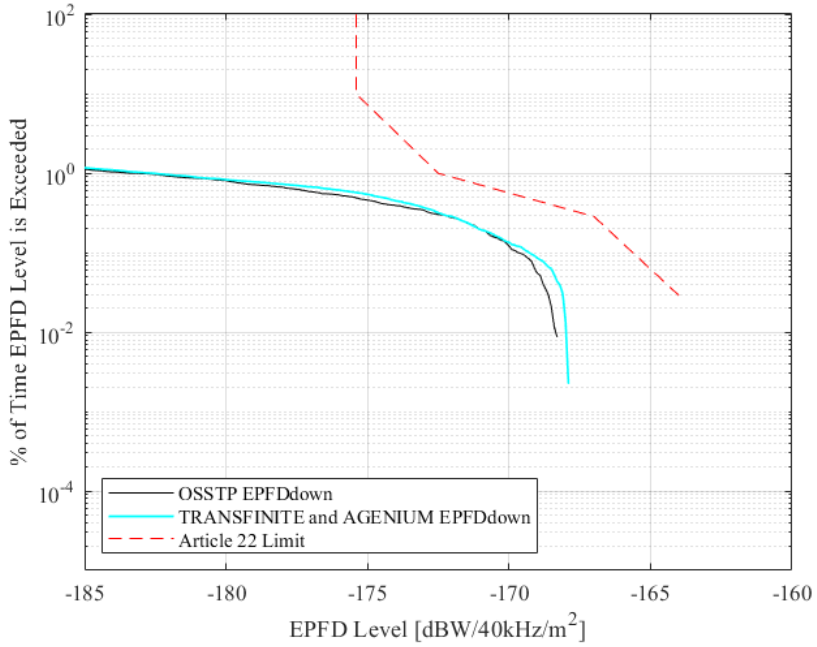

Fig. 13 Validation of METHERA-A's EPFDdown Compliance
This run used a satellite frequency of 17.8 GHz, a dish size of 1m, FSS, and a reference bandwidth of 40kHz.

Both the OSSTP and Transfinite/Agenium results for METHERA-A pass the Article 22 limits as shown in Figure 13. The OSSTP algorithm, Agenium and Transfinite are nearly equivalent in the percent of Time EPFD Level Exceeded. In Figure 14, the constellation 3ECOM-1 is examined. The algorithm examines 12 of its orbital planes containing 24 satellites each. OSSTP's and Transfinite/Agenium's results diverge more concerning this constellation:

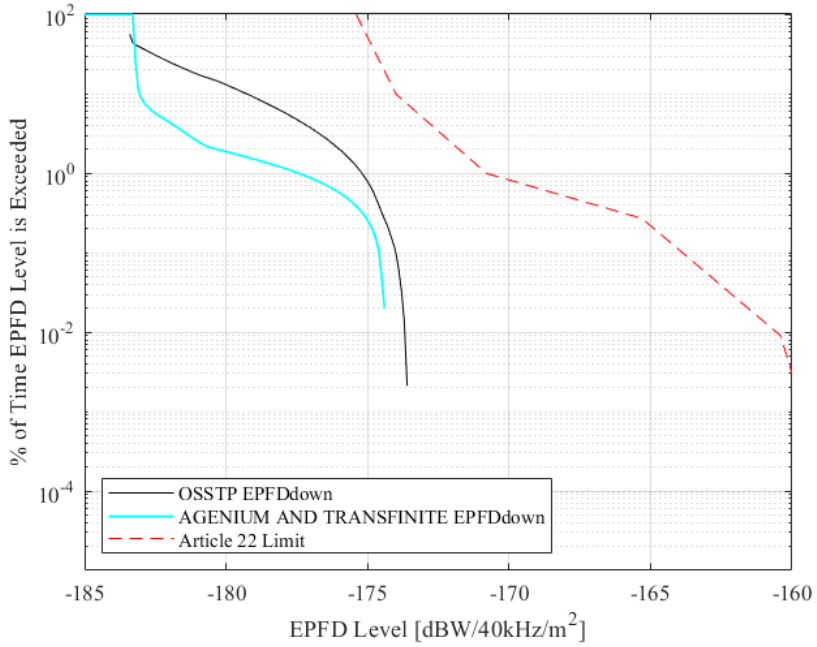

Fig. 14 Validation of 3ECOM-1's EPFDdown Compliance.
This run used a satellite frequency of 10.7 GHz, a dish size of 0.6m, FSS, and a reference bandwidth of 40kHz.

Both the OSSTP and Transfinite/Agenium results pass the Article 22 limits as shown in Figure 14. In Figure 14, however, the difference between the OSSTP and the Agenium and Transfinite results is more noticeable than for ARISTARCHUS's system, shown in Figure 12 and for METHERA-A system, shown in Figure 13. The OSSTP algorithm Percent of Time EPFD Level Exceeded is consistently higher than that of the more conservative Transfinite and Agenium algorithms.

There are multiple possible sources for discrepancy in the OSSTP EPFDdown algorithm as compared to Transfinite and Agenium. Because both Agenium and Transfinite's algorithms are closed source, the methodology both algorithms use to calculate EPFD cannot be verified. OSSTP's differences are likely due to differences in how the α angle is calculated and how EPFDdown values are chosen and summed. Despite these discrepancies, ARISTARCHUS, METHERA-A, and 3ECOM-1's satellite systems were all validated with all three algorithms.

IV. Conclusion

With the deployment of more and more satellite constellations [10], it is essential to ensure the protection of incumbent networks, and coexistence of planned future systems. By quantifying interference and explaining the process in calculating EPFD, more information about protecting such services will be accessible to the public and to satellite operators, which will promote the need for following the EPFD Article 22 limits and ensuring that satellite systems are protected.

Acknowledgements

The authors would like to acknowledge their funding sources: the National Science Foundation (NSF) Center Grant (Federal Grant Number 2037732), Amateur Radio Digital Communications (ARDC), and the Clare Boothe Luce Foundation through the Olin College of Engineering's Clare Boothe Luce Fellowship: an initiative to provide women-identifying researchers funding and support.

Additionally, the authors would like to acknowledge Jamie Santiago and Regan Mah for their work on this paper, Mahima Beltur, Olivia Seitelman, and Utsav Gupta for reviewing this work, along with John Pahl of Transfinite, Mike Lindsay, and Alex Epstein for their mentorship in understanding EPFD.

References

[1] Farahani N. B. and Pouryaie N. "Analysis of Special Interference Methodologies in Satellite Networks". In: (2006), pp. 1–4. DOI: 10.1109/ISAPE.2006.353301.

[2] Transfinite Systems. "Visualyse EPFD". In: (2022). URL: https://www.transfinite.com/content/EPFD.

[3] BR ITU. "User Guide Version 2.3". In: *Equivalent Power Flux-Density Limits Validation Software* (2021). URL: https://www.itu.int/en/ITU-R/space/Documents/EPFDsoftwareUserGuide.pdf.

[4] Hyemi G., Dae-Sub O., and Do-Seob A. "Effective method to assess the impact of interference between Non-GSO system and fixed service". In: (2009), pp. 206–209. DOI: 10.1109/IWSSC.2009.5286395.

[5] ITU. "Article 22". In: *Space services* Chapter VI (1992), pp. 279–298. URL: https://life.itu.int/radioclub/rr/art22.pdf.

[6] ITU. "Recommendation ITU-R S.1428-1". In: *Reference FSS earth-station radiation patterns for use in interference assessment involving non-GSO satellites in frequency bands between 10.7 GHz and 30 GHz* (2000). URL: https://www.itu.int/dms_pubrec/itu-r/rec/s/R-REC-S.1428-1-200102-I!!PDF-E.pdf.

[7] ITU. "Recommendation ITU-R S.1503-3". In: *Functional description to be used in developing software tools for determining conformity of non-geostationary-satellite orbit fixed-satellite system networks with limits contained in Article 22 of the Radio Regulations* (2018). URL: https://www.itu.int/dms_pubrec/itu-r/rec/s/R-REC-S.1503-3-201801-I!!PDF-E.pdf.

[8] ITU. "Recommendation ITU-R S.1503-2". In: *Functional description to be used in developing software tools for determining conformity of non-geostationary-satellite orbit fixed-satellite service systems or networks with limits contained in Article 22 of the Radio Regulations* (2013). URL: https://www.itu.int/dms_pubrec/itu-r/rec/s/R-REC-S.1503-2-201312-S!!PDF-E.pdf.

[9] ITU. "Recommendation ITU-R BO.1443-3". In: *Reference BSS earth station antenna patterns for use in interference assessment involving non-GSO satellites in frequency bands covered by RR Appendix 30* (2013). URL: https://www.itu.int/dms_pubrec/itu-r/rec/bo/R-REC-BO.1443-3-201407-I!!PDF-E.pdf.

[10] Kriezis A., Agarwal R., Lisy C., Seitelman O., Mah R., Gupta U., and Lohmeyer W. Q. "Power Flux Density (PFD) Compliance Validation of FCC's Ka-band NGSO Processing Round Participants". In: *2021 IEEE Aerospace Conference (50100)* (2021). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9438491&tag=1.

Part V

Searching for Correlations Between Lunar Crater Features and Location with Machine Learning

Searching for Correlations Between Lunar Crater Features and Location with Machine Learning

Braden Oh* and Michael Remley†
Olin College of Engineering, Needham, MA, 02492

Modern machine learning algorithms may be employed to search for correlations between the properties and location of lunar craters. This study investigates the possibility of using machine learning classifiers to identify new patterns that legacy data could not reveal by training classifiers on a database released by Dr. Stuart Robbins in 2018, the most complete lunar database to date. Two types of classifiers, random forest and multi-layer perceptron, were each trained on twin datasets of craters grouped by location, using either USGS quadrangle or k-means clustering. Following training, the models were presented with a set of craters not present in the training data for which all data in the Robbins database except for latitude and longitude were provided. Both classifiers performed significantly better at classifying craters than random guessing when using quadrangle-grouped data; however, both classifiers placed the highest weight on the longitudinal error in ellipse fits, which is correlated with latitude. As a result, we conclude that machine learning classifiers are not able to easily uncover patterns between crater geology and lunar location due to the strongest predictors of crater location not being intrinsic features of the craters themselves.

I. Introduction

One of the great stories humanity seeks is that of its creation. This inquiry about the origin of our universe, our planet, and ourselves is augmented by studying the formation of the Moon. Solar system formation is a chaotic process, meaning it is difficult to reconstruct the history of planetary formation only from physical laws and known conditions [1, 2] . Unable to extrapolate, we directly study planetary formation and assume that the same processes which formed one body in the solar system were also formed the others [2]. With this assumption, our study of the Moon provides insights into various planetary formation processes, including those that cause: surface feature formation [3]; variations of isotopes observed in geology [4]; results of enormous planetesimal impacts [5]; and dry landslides on Mars [6]. This helps us build a chronological story of events in the early solar system [7] and allows us to reconstruct our story of creation. This narrative is scrawled (at least in part) upon the surface of the Moon - further investigation of which provides clues to our origin story.

Robbins [8] accompanied the release of the Robbins Lunar Crater Database to describe the methodology, limitations, and possibilities of the most complete lunar database to date. Crater data were derived from manually selected rim points on images from the Lunar Reconnaissance Orbiter (LRO) Camera's wide-angle camera (WAC), the LRO's Lunar Laser Altimeter (LOLA), and the Terrain Camera (TC) on SELENE/Kaguya. The database contains position, orientation, circle fits, ellipse fits, fit quality, and the number of points used to identify each crater. Multiple sources of random variation between 1 and 10% are present from projection errors and human variation. Robbins' ellipticity and spatial density analyses are consistent with previous databases, and new trends were not immediately apparent. As perhaps the most complete database to date, Robbins emphasizes possibilities for future work to spot patterns where legacy data could not reveal any. We ask the question of whether machine learning algorithms can identify patterns between crater features and location on the moon, seeking to uncover some of these patterns into which Robbins emphasized future research.

*Engineering Physics, class of 2023
†Electromechanical Engineering, class of 2022

II. Methods

A. Data Pre-Processing

We took the database from [8] and performed three pre-processing steps before using the data to train a pair of machine learning models. We began by removing any craters containing a *NaN* (not a number) value. This required us to remove eight craters; being that the entire database contains 83,061 craters, the removed values compose less than 0.01% of the database. Next, we grouped the craters into thirty regions (a detailed description of this process and the reasoning behind it is given in Section II.B). Each crater was assigned to a region represented by an integer ranging from 0 to 29, inclusive. This region index was added to each crater in the database, represented as a new feature column. Our final pre-processing step was to remove exact location information and irrelevant data from the database. We removed six pieces of information from each crater: 'Unnamed: 0', 'CRATER_ID', 'LAT_CIRC_IMG', 'LON_CIRC_IMG', 'LAT_ELLI_IMG', and 'LON_ELLI_IMG.' The first two columns removed are index columns that provide no information about the crater features, so serve no purpose in the machine learning training process. The latter four columns contain the latitude and longitude coordinates for the centers of a circle and ellipse fit to each crater. These location coordinates are removed because a crater's location is sufficiently captured by its assigned group and because allowing exact coordinates to remain would serve as a "dead giveaway" to a machine learning algorithm seeking to predict crater location by features alone.

B. Region Assignment

We used two region assignment methods: the United States Geological Service (USGS) lunar quadrangle system and k-means clustering. Each crater is labelled according to the region in which it lies so that a machine learning algorithm can predict the label again later given solely non-positional data. Both of these region labelling methods add a new column to the lunar crater database called "LABEL" which stores an integer index for each crater region.

1. USGS Quadrangles

The USGS has divided the moon into thirty quadrangles at the 1:2.5 million scale. The division gives regions that are more equal in area than if the latitude and longitude slices were all equal intervals. The USGS labels these Lunar Quadrangles (LQ) regions from 1-30 as "LQ01" to "LQ30", but we have labelled them with integers 0-29 where 0 maps to LQ01, 1 to LQ02, and so on, because computers index from 0. Figure 1 depicts this subdivision on a latitude vs. longitude graph of lunar crater positions. Observe that certain regions, such as 10, are less dense than others. This uneven distribution is quantified in the corresponding histogram in Fig.1.

2. k-Means Clusters

k-means clustering is an algorithm that seeks to group data into k clusters. This results in the distribution of craters in each region being more even than in the quadrangle grouping. The k-means algorithm was run on 'LAT_CIRC_IMG' and 'LON_CIRC_IMG', the latitude and longitude coordinates of the center of each fitted circle for $k = 30$, that is, to cluster the data into 30 regions.

The clustering algorithm begins by selecting k random points to use as cluster centroids. The clustering algorithm then alternates between two steps: *assignment* and *update*. During the assignment step, each crater is assigned to the cluster with the nearest centroid. During the update step, the mean of each cluster is calculated and the centroid is moved to that mean location. The algorithm then re-assigns the craters to these new centroids and the algorithm repeats until the craters are consistently re-assigned to the same cluster. Because the initial points are randomly selected, the algorithm is not guaranteed to find the optimal solution, just a converged solution. For this reason the craters are not equally grouped in each region, but are significantly better distributed than in the quadrangle regions.

To select the number of clusters we would use, we swept a range from 15 to 45 clusters, evaluating a single decision tree classifier (a detailed description of decision tree classification is provided in section II.C.2). We discovered that decreasing the number of clusters (and thus reducing the number of labels for a classifier to select from) increased the accuracy of the decision tree but also increased the probability of success by random guessing. Discerning a negative correlation between accuracy and number of clusters, we decided to use 30 clusters so that we could easily compare the results from training on the k-means clustered craters to the results from training on the quadrangle-grouped craters.

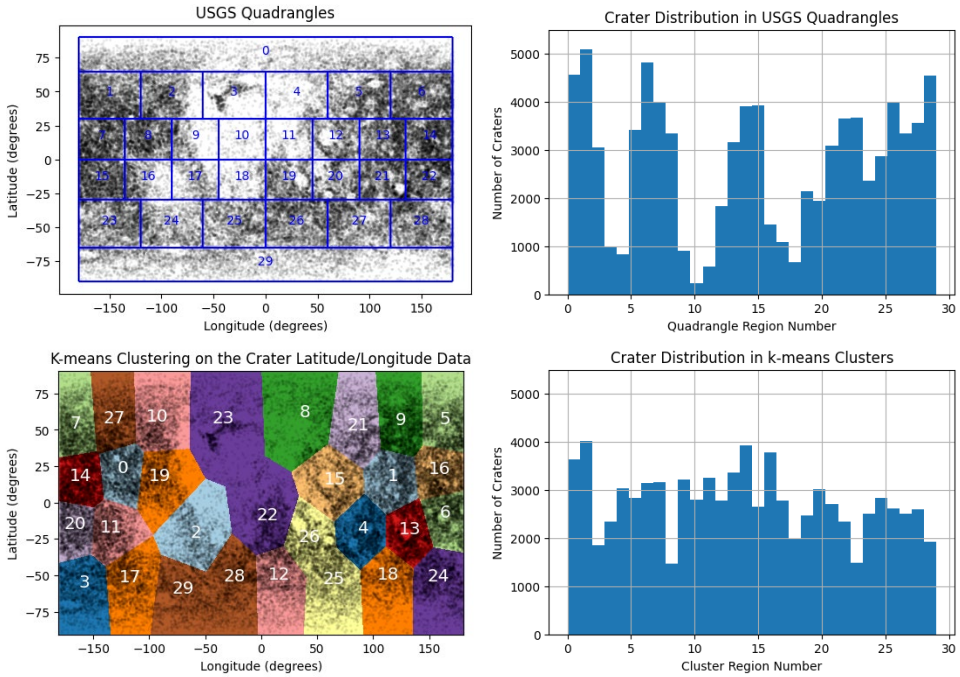

Fig. 1 (Top left) Map of lunar crater positions with USGS quadrangle overlay. Crater centers plotted with 10% transparency. (Top right) Distribution of lunar craters in USGS quadrangle regions. This distribution could bias the algorithms to classify craters into certain regions simply because they have more craters. If this bias is present, regrouping our data more evenly would significantly change the error rate. (Bottom left) Map of lunar crater positions regrouped by k-means clustering. Crater centers also plotted with 10% transparency. (Bottom right) Distribution of lunar craters in 30 k-means regions. These regions are more evenly distributed and consistently produce higher error rates than the USGS regions.

C. Classifiers

We tested two different *classifiers*, algorithms which take some input data and sorts them by group. Both of our classifiers take the 16 non-positional data columns as input and classify the crater into one of 30 regions. A randomly guessing classifier would guess correctly a mere 3.33% ($1/30^{th}$) of the time. Since guessing correctly is incentivized, unequal distributions present a problem where accuracy is increased by guessing regions that have more craters. Two ways of handling this are to train classifiers on an equal-distribution subset of craters or to have equally-distributed craters to begin with. The USGS quadrangles distribution presents the unequal distribution problem, so we introduced k-means clustering to generate a more equally distributed dataset.

Classifiers are a powerful tool to automatically recognize patterns where the human eye may have overlooked them. If the classifiers are consistently good at recognizing some craters more than others, then the well-recognized craters must be distinct in a measurable way because this would not happen consistently by chance.

1. Data Splitting

Classifiers need to be tested on data which is not included in the training set. The training data and testing data must have no overlap but should be statistically similar. To determine what percentage of data we would reserve for testing, we experimented on single decision tree classifiers at a time (a detailed description of decision tree classification is provided in section II.C.2), sweeping a range from 1% to 40% of data reserved for training. We discovered that reducing the data provided for training made little difference until around 25%, when the accuracy of the decision tree began to gradually decline. We concluded that little advantage could be gained by reducing the training set and so decided to use

99% of the data (82222 craters) for training and reserved only 1% (831 craters) for testing.

2. Decision Trees and the Random Forest

The decision tree is a classifier that makes a series of binary decisions in order to predict a target value for a data point. In the context of craters, a training set of crater features along with a list of correct location labels is fed into a decision tree model, and the model builds a 'tree' of if-then statements that begins with a single switch point and ends with a location label. For an unknown crater, the first switch is checked and then the data point is passed to a subsequent switch and so forth until finally reaching a label. The various crater features are weighted by importance and appear in the switch hierarchy accordingly.

One primary advantage of the decision tree over other machine learning algorithms is that it can be easily understood. The tree can be conceptualized in a simple way and the importance of particular attributes is directly reflected in the hierarchy of switch statements. Another advantage is that the tree is robust to input data, requiring no normalization or scaling of the input data prior to training. A primary disadvantage of the decision tree is that it is highly sensitive to changes in the training data. Small changes in training data can result in large changes to the final tree, meaning two randomly extracted test sets may result in two classifiers that differ in accuracy by entire percentage points. Furthermore, the training process is susceptible to over-fitting the data, resulting in decision trees that are highly complex without being more accurate.

One way to combat the limitations of the decision tree is to train an ensemble of decision trees and use the output of each tree in a vote to produce a final classification. Individual trees are trained on subsets of the training data, so vary in their decision processes and accuracies. This is known as a *random forest* and is effective at resisting the tendency of trees to over-fit the data. We found that training a forest of 100-200 trees led to reliably better classifications than were produced a single decision tree, but training such a forest required significant computational resources.

3. Multi-Layer Perceptron

A perceptron is a simple neural network with an input layer with one neuron for every input and an output layer with a neuron for every output. Every input neuron is connected to every output neuron, and the weight of each connection determines how much activation at the input neuron is forwarded to the output. A multi-layer perceptron (MLP) introduces one or more hidden layers between the input and output layers to allow the network to learn complex behaviors. In either case, the entire network starts with random-valued weights and is trained on a set of data by incentivizing weights that tend to produce the correct output. In our case, we have 16 input neurons for each of the 16 data columns, 20 neurons chosen in a single hidden layer, and 30 neurons in the output layer for each of the 30 regions. Whichever output neuron activates the most indicates the region the perceptron thinks the column came from.

We present two perceptrons, one for each type of region classification. Both have the same structure and are trained and tested in the same way (10% of data used for testing). Unlike the random forest classifier, the MLP classifiers require data normalization to fall within a range of 0 to 1 within every column in the database. The final relative scaling of the weights on the input layer indicate which data columns are most important in determining a crater's region, and some guesses can be made about combinations of parameters by examining the hidden layer. Features that are important to both the perceptrons and random forests may reveal new patterns in the data.

III. Results

A. The Confusion Matrix

A convenient way of gaining insights into where a classifier makes errors is via a visual representation known as a *confusion matrix*. A confusion matrix is an $n \times n$ table with rows that represent the true label of a data point and columns that represent the model's predicted label. Each cell contains an integer which represents the number of craters which the model classified in that way. Scikit Learn [9, 10] provides a built-in method for plotting confusion matrices which represents the predicted labels in ascending order along the x-axis and the true labels in descending order along the y-axis. For each cell, the number inside represents the number of craters with an actual label, y, to which the classifier attached a predicted label, x. Thus, in this representation, the cells which correspond to correctly predicted craters fall along the main diagonal which runs from top left to bottom right. Figure 2 shows the confusion matrix for a highly accurate random forest. The model which produced that matrix was trained on a dataset that included the true labels, so the high accuracy is no surprise.

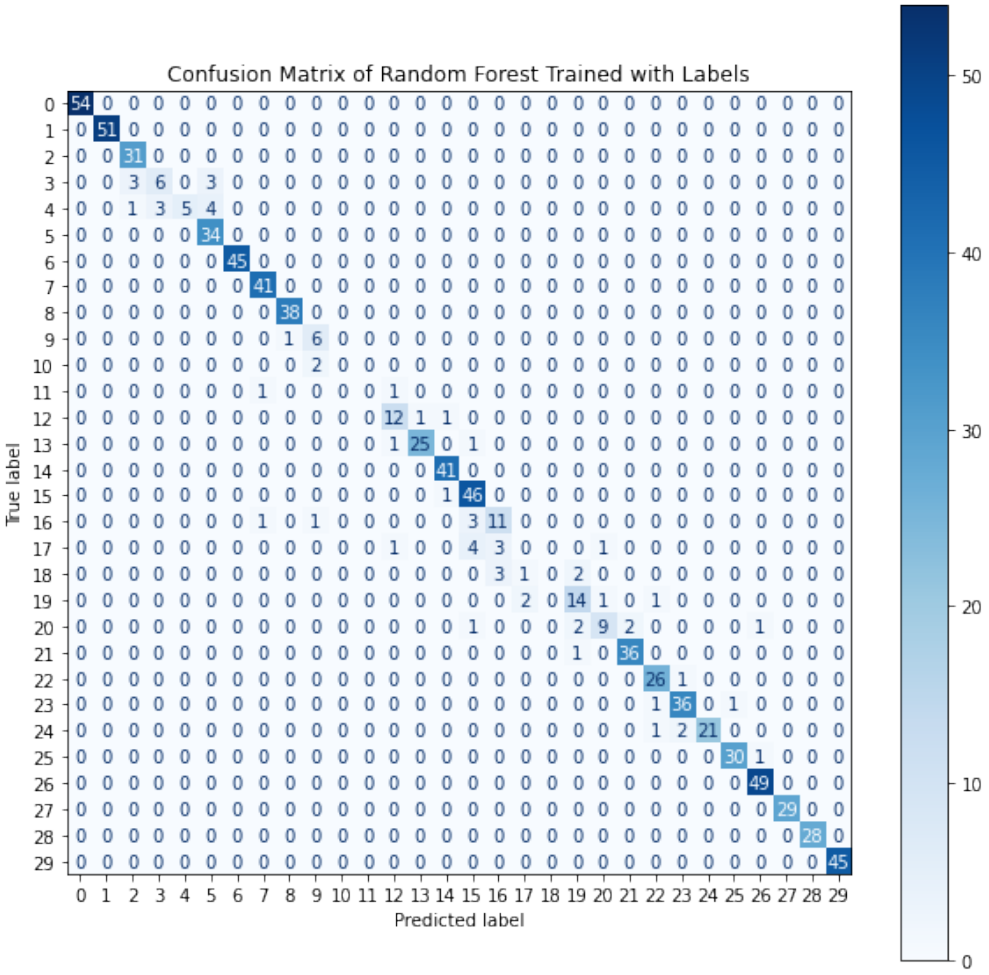

Fig. 2 The confusion matrix from a random forest trained with the location labels present in the training set. This forest has an accuracy of 92.5% at identifying craters in the test set and the high accuracy is visible in the diagonal line across the matrix. The Y-axis of the confusion matrix shows the true location of each crater while the X-axis shows the location predicted by the random forest. Each integer label corresponds to a USGS quadrangle.

B. Decision Trees and the Random Forest

We began by training and testing a variety of single decision trees on data grouped by k-means clustering and found accuracies that generally fell within the range 7.5-9%. We then performed an identical set of tests on data grouped by quadrangle and discovered that the accuracies averaged significantly higher, generally falling within 11-13% accurate. Repeated testing proved this disparity between groupings to be consistent, so we decided to investigate further as we moved ahead to training a random forest.

Training random forests proved to be highly computationally intensive for large numbers of trees, so we swept a range of tree numbers to find out whether accuracy was truly a function of forest size. We trained forests that ranged from 50 to 200 trees on data grouped by both quadrangle and clustering. We discovered that for forests larger than 100 trees, accuracy increased with additional trees, but the gains were marginal against rapidly increasing training time and model size (a forest of 200 trees took nearly 3 minutes to train and produced a model that consumed over 4GB of memory). As in previous tests, grouping the data by quadrangle produced a higher accuracy than by clustering, but the disparity was far larger in a random forest: clustered grouping consistently produced results over 10% less accurate.

Ultimately we decided to settle on 100 trees for our forest size, requiring a reasonable amount of training time and disk space. We trained 25 forests of this size on the data grouped by USGS quadrangle, with identical training and

testing sets, and recorded the accuracy of each. These forests had accuracies that ranged from a minimum of 16.25% to a maximum of 18.29%, with a mean of 17.09% and a standard deviation of 0.65%. Thus, these random forests performed, on average, 5.13 times better than random guessing. After gathering these data we continued to train individual forests with randomly extracted training sets to find out whether one could exceed 18.29% accuracy. Ultimately, we managed to train a forest with 19.25% accuracy, from which we extracted the importance of each feature in the training set and produced a confusion matrix. These two graphs are shown in Fig. 3.

The feature weighting graph provides insight into which features included by Robbins proved most significant in classifying the craters. The feature of greatest importance is 'LON_ELLI_SD_IMG,' carrying 26.80% of the weight of classification. This parameter is described by [8] as the "Formal standard error in the ellipse fit's center longitude, in degrees." This error increases in part because an slight longitudinal distance error in distance units will produce a higher longitudinal error in degrees near the poles. As a result, this column is a proxy for latitude since it would be highest at high latitudes, even if the distance error is constant for all craters. The decision tree seems to have spotted this and uses it to identify polar craters.

The confusion matrix yielded a light diagonal line, indicating its successes. The matrix also provided the insights that the random forest struggled to tell apart craters in regions 0 and 29 and in regions 21 and 22. Regions 0 and 29 correspond to the poles of the moon, indicating that the random forest struggled to tell apart craters at the north and south poles, while regions 21 and 22 are adjacent regions that are fairly crater-dense. The random forest also tended to over-assign craters to regions 1 and 6. These regions (which correspond to USGS quadrangles LQ02 and LQ07, respectively) are adjacent regions at northern latitudes that are crater-dense and enclose very little mare. Furthermore, the random forest performed very well in region 16.

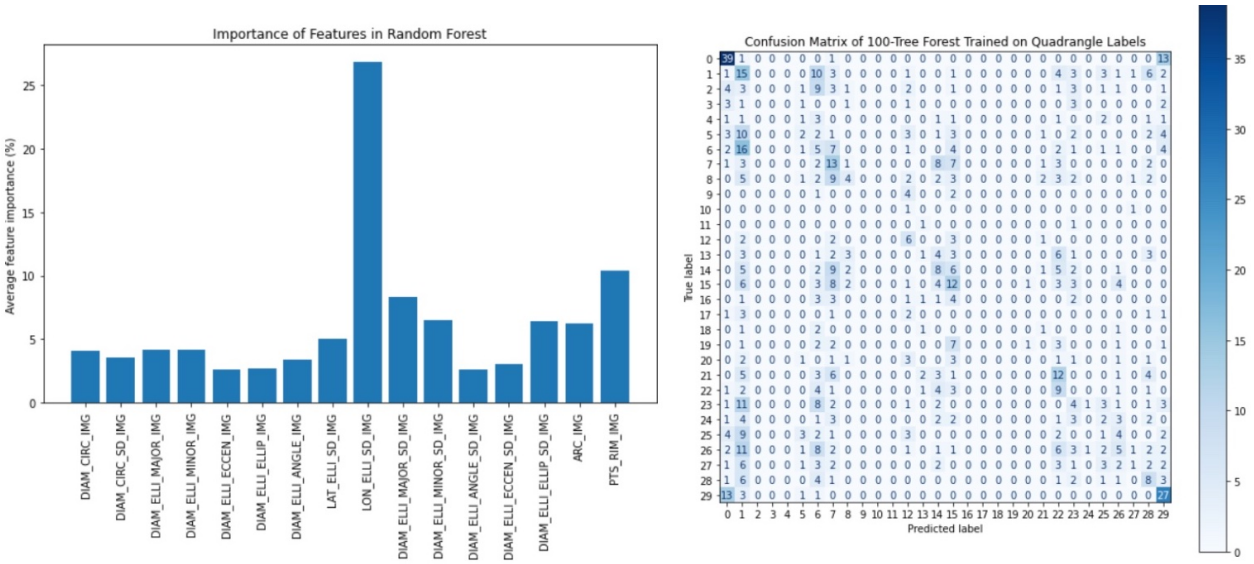

Fig. 3 (Left) The 16 crater features used for training and their relative importance in a random forest of 100 decision trees. Each tree has its own weighting of feature importance; this graph shows the average importance of each feature, taken as an arithmetic mean across all 100 trees in the forest. (Right) The confusion matrix for the same forest. This random forest has a classification accuracy of 19.25%. Higher numbers of predicted craters are represented by darker squares. The true location is shown on the y-axis and the predicted on the x-axis. A perfect prediction would result in a diagonal line from top left to bottom right; the faint diagonal line in this figure is an indicator of the algorithm's successes.

C. Multi-layer Perceptron (Neural Network)

The accuracy of the multi-layer perceptron (MLP) was around 8-14% when we first trained on both k-means and USGS regions. We discovered the MLP was bottle-necking through a 2-neuron hidden layer, significantly hindering its performance. After retraining both models, we arrived at the results depicted in Fig. 4 and Fig. 5. The accuracy of the k-means MLP was 13.6%, much worse than the 20.0% achieved by the USGS MLP but still better than the baseline

3.33% chance from random guessing. This decrease in performance is likely due to the undermined strategy of guessing the most populated regions disproportionately more. This explanation is further supported by the observation that the k-means MLP predicted all but one region at least ten times, while the USGS MLP failed to predict 9 regions a meanningful number of times. Although further from the ideal diagonal matrix, the k-means MLP may be more likely to have identified meaningful characteristics of the dataset than the USGS MLP.

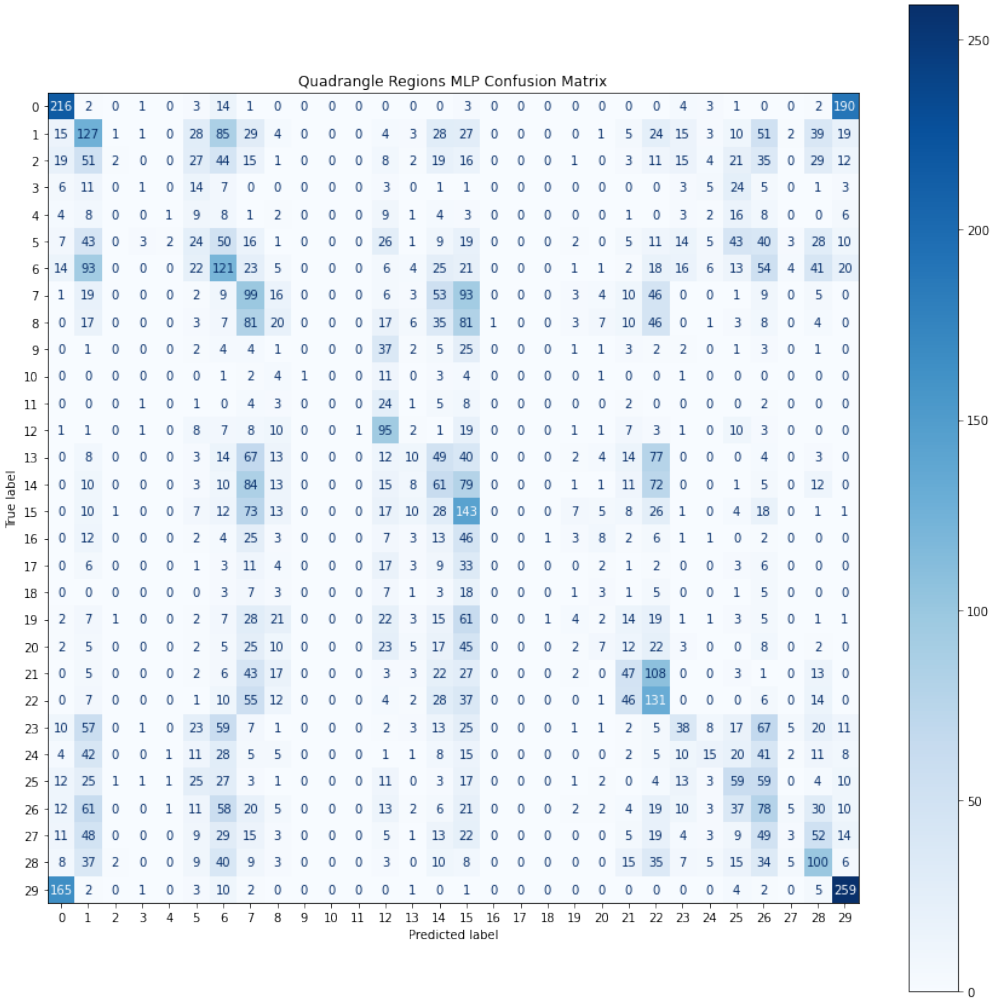

Fig. 4 The confusion matrix for the multi-layer perceptron using crater data categorized into USGS regions. Polar craters are usually correctly identified as polar, though the classifier confuses the poles nearly half of the time. Regions such as 0, 1, 15, 22, and 29 are guessed most often because they were represented the most in the training data.

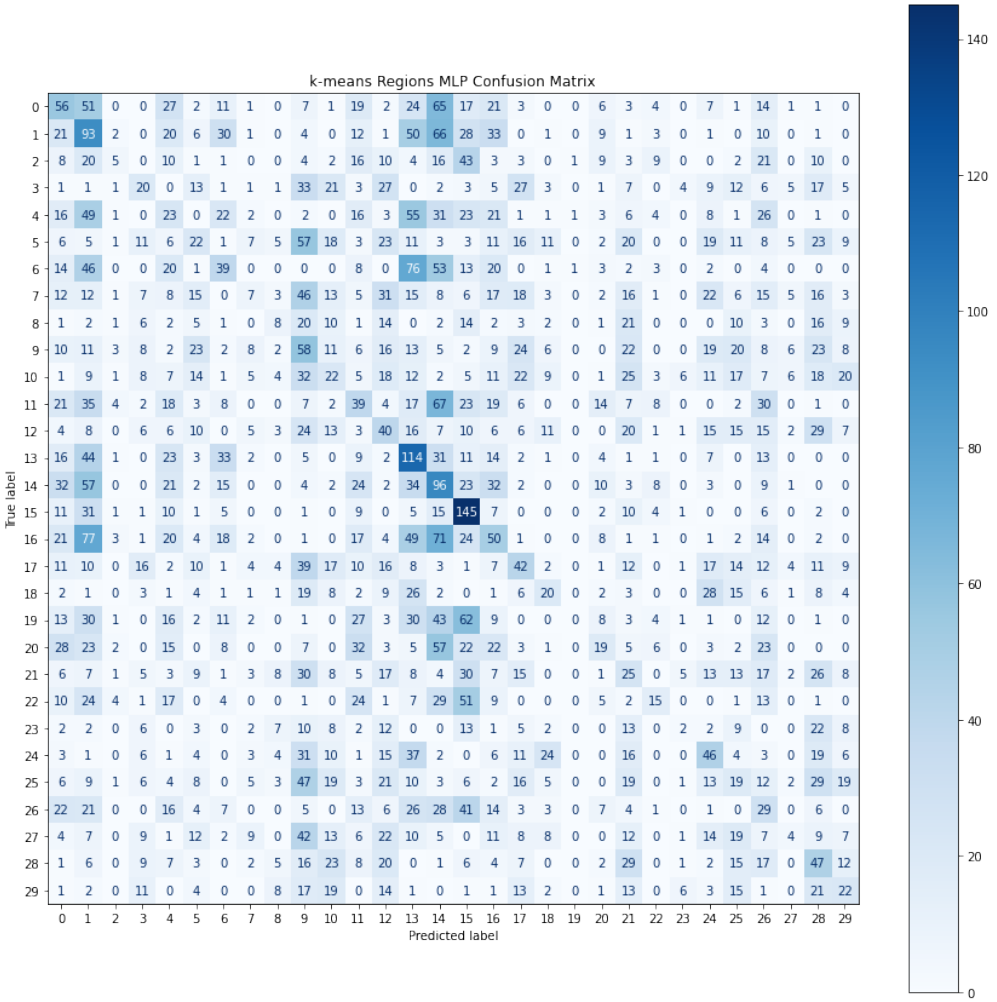

Fig. 5 The confusion matrix for the multi-layer perceptron using crater data categorized into regions found by k-means clusters. Overall accuracy is lower and the confusion is more diffuse across the whole matrix, but region 15 seems to stand out as one that is guessed most often. This region happens to be the Sea of Tranquility.

In Fig. 4, the corners of the matrix show that craters are accurately identified as belonging to one of the two poles, perhaps because the uncertainty in position is highest for these craters. The classifier also confuses north polar craters with south polar craters often- and vice versa. Other regions, such as region 1 (LQ02) and 15 (LQ16), were inaccurately predicted more often than the true number of craters in each, a sign that over fitting may be occuring in the USGS MLP. The results in Fig. 5 are more evenly distributed and lack the polar confusion since the poles are divided among several regions. Interestingly, 50% of craters belonging to region 15 were correctly classified. This area corresponds to the Sea of Tranquility.

We also depict the weights of the MLPs in Fig. 6 and Fig. 7 . From the relative sizes of the weights depicted by their thickness, we observed that in both networks, the uncertainty in position and uncertainty in the major axis of an ellipse fit of craters were the most important determinants of the region in which they would classify. The next most important parameters are related to the size of the crater, including the circular fit diameter and points used for the fit. The importance of uncertainty in longitude from the ellipse fit (LON_ELLI_SD_IMG) is consistent with the decision tree classifier. In Fig. 8, we can see that the longitude standard deviation in the ellipse fit is indeed related to whether a crater is close to the poles or not in the original lunar database. Other agreements between the two classifiers are less common; the decision tree classifier lists PTS_RIM_IMG as second-most important while neither MLP classifier does so.

In summary, the accuracy of the random forest classifier was 19.25% with quadrangle regions and 3.13% with

clustered regions. For the MLPs, the accuracy was 20.0% with quadrangle regions and 13.6% with clustered clusters. This significant performance suggests a strong correlation between crater features and location on the Moon. Patterns exist between intrinsic crater features - such as ellipticity, size, and measurement accuracy - and location on the Moon. This is an encouraging sign for machine learning's ability to uncover hidden patterns within lunar crater data and opens a door to future geologic analysis with modern machine learning tools.

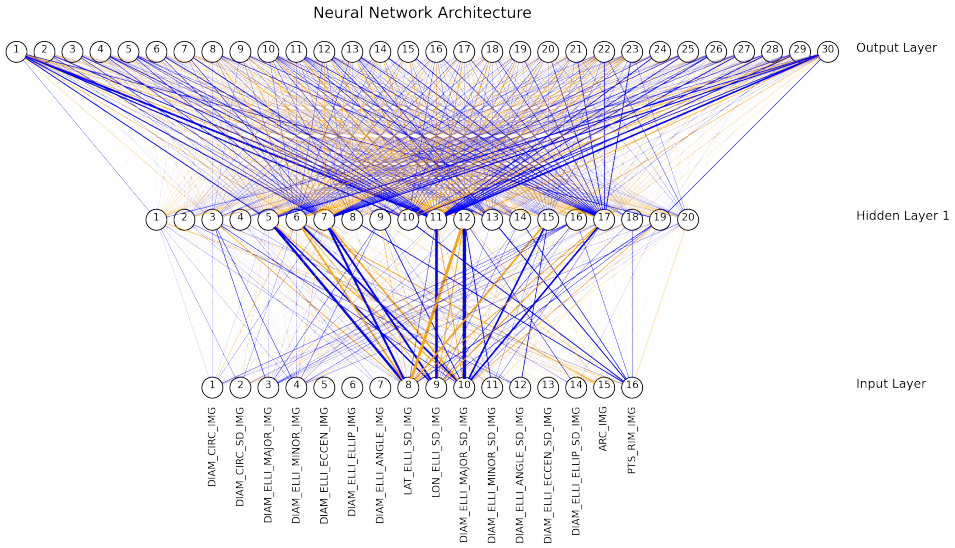

Fig. 6 The network structure for the MLP trained on USGS quadrangle data. Blue weights are positively-valued, and orange weights are negatively-valued. The size of the lines representing each weight indicate their relative weighting. From the strong weights associated with LAT_ELLI_SD_IMG, LON_ELLI_SD_IMG, and DIAM_ELLI_MAJOR_SD_IMG, we infer that these parameters are most important for classification. Created with Liu [11].

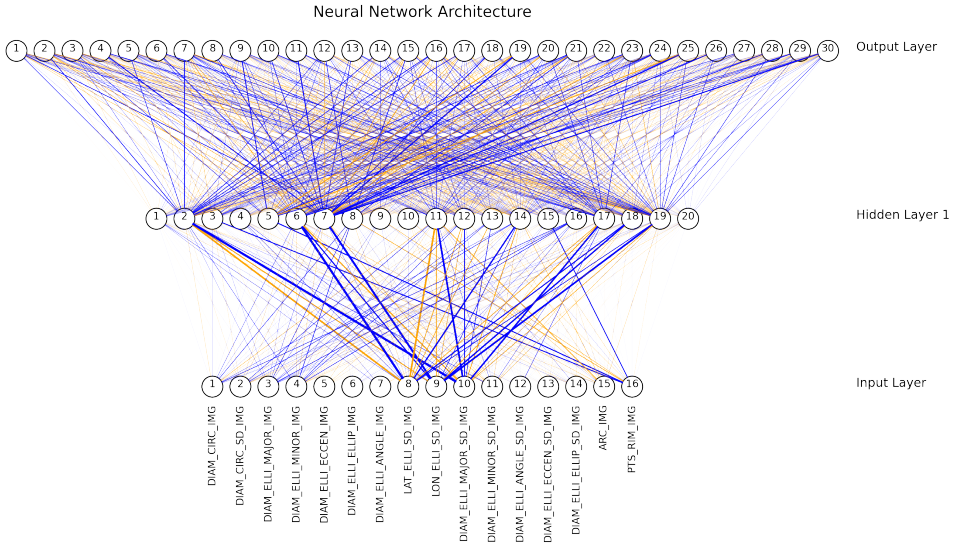

Fig. 7 The network structure for the MLP trained on k-means cluster data. Created with Liu [11].

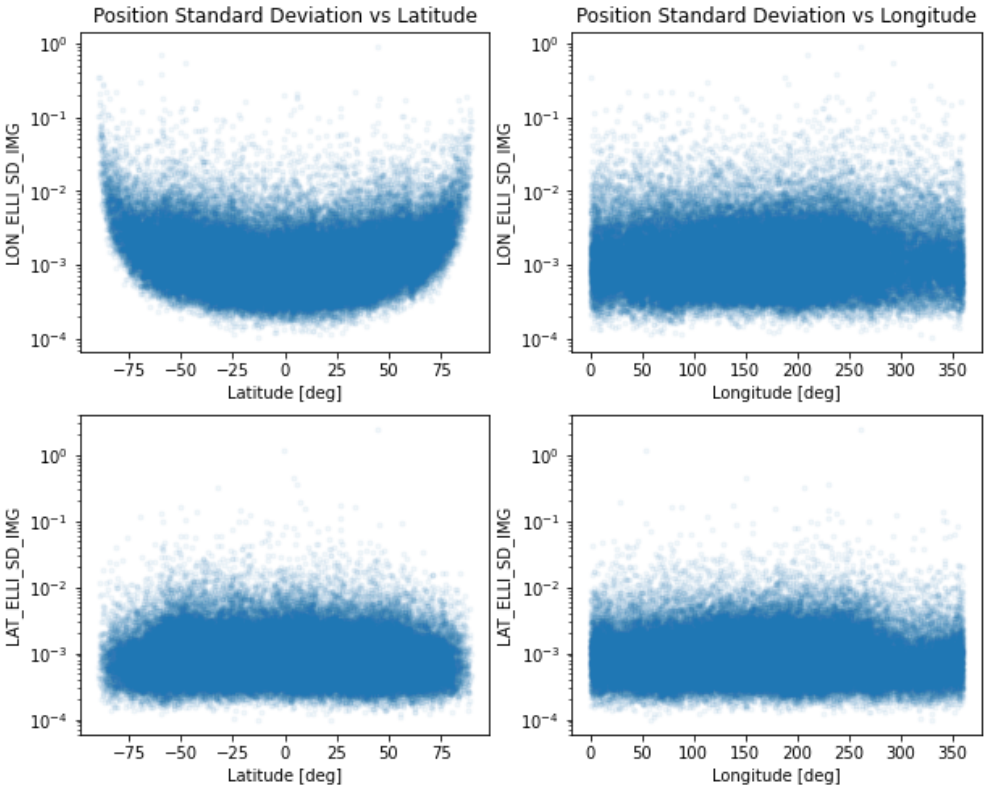

Fig. 8 Plots of position standard deviation vs position for ellipse fits. Longitudinal standard deviation increases near the poles because the degree error per unit distance at high latitude is greater than the degree error near the equator. The classifiers both identified this as a way to recognize polar craters.

IV. Conclusions

Both the random forest and the multi-layer perceptron classifiers identified the longitudinal error in ellipse fits as a strong predictor of latitude, correctly identifying many craters near the poles. Beyond this longitudinal error the classifiers diverged on the weighting of other parameters, with the random forest next favoring the number of rim points, and the perceptrons favoring ellipse fit major axis standard deviation. In conjunction with the longitudinal error as a predictor of latitude, the random forest and perceptrons both achieved higher accuracy with data grouped by USGS quadrangle as opposed to k-means cluster. We believe that this is due to the fact that both classifiers most highly weighted the database feature 'LON_ELLI_SD_IMG.' The graph of this feature against the latitude, as shown in Fig. 8, shows a strong indication that polar craters do not have low errors and indicates that latitudes very near ±90° have high errors. With the USGS quadrangles, only two regions correspond to the poles (indices 1 and 29, the north and south pole, respectively), allowing the classifiers to decide between only two regions for classification of a polar crater. With the clustered data, however, the polar craters are distributed across many more regions, making classification into a specific region more difficult for a classifier. The final accuracy of the random forest classifier was 19.25% with quadrangle regions and 3.13% with clustered regions. For the perceptrons the accuracy was 20.0% with quadrangle regions and 13.6% with clustered clusters. Both quadrangle groupings performed nearly six times better than would be achieved by random guessing. Overall, however, the strongest predictors of crater location were found to be errors in the measured position and size of craters, not intrinsic features of the craters themselves. This indicates that machine learning was not able to uncover patterns between crater geology and lunar location, so our overarching question about machine learning's potential to contribute to lunar geology and crater study remains unanswered.

Acknowledgments

This work was performed as a class project for the Spring 2020 section of the course Astronomy & Statistics. We thank Prof. Carrie Nugent for all of the effort she put into adapting this course for remote learning as a result of the the COVID-19 pandemic. We also thank the editors and reviewers of the Olin Undergraduate Research Journal for the work they put into establishing this new journal.

References

[1] Whipple, F., "The History of the Solar System," *Centennial: First Scientific Session*, Vol. 54, 1964, pp. 565–594. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC300311/pdf/pnas00182-0369.pdf.

[2] Shoemaker, E., *Interpretation of Lunar Craters*, No. Chapter 8 in Physics and Astronomy of the Moon, Academic Press Inc., 1962. URL https://books.google.com/books?hl=en&lr=&id=A883BQAAQBAJ&oi=fnd&pg=PA283&dq=lunar+craters&ots=cm8_UGrnwr&sig=llaCjlSnZ3vxQDXYIuvowi8f28A#v=onepage&q=lunar%20craters&f=false.

[3] Singer, S., "The early history of the earth—moon system," *Earth-Science Reviews*, Vol. 13, No. 2, 1977. URL https://www.sciencedirect.com/science/article/abs/pii/0012825277900216.

[4] Halliday, A., "Terrestrial accretion rates and the origin of the Moon," *Earth and Planetary Science Letters*, Vol. 176, No. 1, 2000. URL https://www.sciencedirect.com/science/article/abs/pii/S0012821X99003179.

[5] Hartmann, W., "Moon Origin: The Impact-Trigger Hypothesis," *Conference Proceedings*, 1986. URL http://adsabs.harvard.edu/pdf/1986ormo.conf..579H.

[6] Bart, G., "Comparison of small lunar landslides and martian gullies," *Icarus (Elsevier)*, Vol. 187, No. 2, 2007. URL https://www.sciencedirect.com/science/article/abs/pii/S0019103506004106.

[7] Hiesinger, H., van der Bogert, C. H., and Pasckert, J. H., "How old are young lunar craters?" *Journal of Geophysical Research*, 2012. URL https://doi.org/10.1029/2011JE003935.

[8] Robbins, S., "A New Global Database of Lunar Impact Craters >1–2 km," Vol. 124, 2018, pp. 871–892. https://doi.org/https://doi.org/10.1029/2018JE005592.

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[10] INRIA, "scikit-learn," , 2021. URL https://scikit-learn.org/stable/.

[11] Liu, J., "visualize-neural-network," , 2018. URL https://github.com/jzliu-100/visualize-neural-network.

Part VI
Untangling the Neural-Fly Code

Untangling the Neural-Fly Code

Isabel de Luis
Olin College of Engineering, Needham, MA, 02492
Calfornia Institute of Technology, Pasadena, CA, 91125

Currently, most uninhabited aerial vehicles (UAVs) use linear control, and while they have been taught to fly impressive demos such as formation flying, these flights occur under ideal conditions. Neural Fly is a deep-learning method that allows UAVs to adapt to real-world wind conditions, enabling more accurate control than traditional methods. There are three prongs to Neural Fly—the hardware, the software in the loop (SITL) simulator to test the hardware virtually, and the code simulator. However, the code simulator is written in Python while the hardware and SITL simulator are operated with ROS in C++. While prototyping is done in Python due to the ease of development, the vehicle runs on C++ due to the performance benefits. Thus, the code developed for the simulator cannot directly interact with the SITL and hardware. This paper discusses how to integrate Python and C++ to use the controller for the code simulator in the SITL and hardware, removing the need to reimplement the controller developed in Python in C++ for deployment.

I. Introduction

Currently, most drones are flown in controlled conditions, and while they have been taught to fly in formation, these flights usually occur under ideal conditions [1]. One of the challenges facing autonomous uninhabited aerial vehicles (UAVs) is flying in real weather conditions. With the commoditization of UAVs, control of these vehicles is required to become more precise and agile.

A. Neural-Fly

Neural-Fly allows drones to adapt to real-world weather conditions by incorporating common representations through deep learning. Aerodynamics in different wind conditions share a common representation and the wind-specific part lies in a low-dimensional space. Neural-Fly builds on these two observations by using a domain adversarially invariant meta-learning (DAIML) algorithm to learn the share representation. With this basis, Neural-Fly uses a composite adaption law to update a set of linear coefficients for mixing the basic elements. Neural-Fly was tested at the Caltech Real Weather Wind Tunnel, and then outside, where it achieved precise flight control [1]. Neural-Fly uses a position-integral-derivative (PID) controller as a baseline controller to control a UAV's roll, pitch, yaw, and thrust. Additionally, Neural Fly augments the PID controller with the forces learned via machine learning which improves tracking accuracy. Essentially, Neural Fly's PID controller uses the UAV's position, velocity, and acceleration to calculate the desired roll, pitch, yaw, and thrust by calculating the proportional, integral, and derivative responses and summing up those components [2]. Roll, pitch, and yaw are the rotational axis of an UAV, where roll is rotation about the x-axis, pitch is rotation about the y-axis, and yaw is rotation about the z-axis [3].

B. Project Outline

Neural-Fly has three prongs to it—the simulator to test the code, the software in the loop (SITL) simulator to test the code on the hardware, and the hardware used to conduct the experiments. However, the simulator cannot interact with the SITL or hardware because the simulator is written in Python while the SITL and hardware run in C++. This disconnect makes it impossible to use the same controller in all three aspects of Neural Fly. The current implementation thus has two controller codes–a Python controller to use in the simulator and a C++ controller to use in the SITL and hardware.

The goal of the project discussed was to create a way for the simulator controller to be used on the SITL and hardware so that researchers do not have to implement the same code in both C++ and Python.

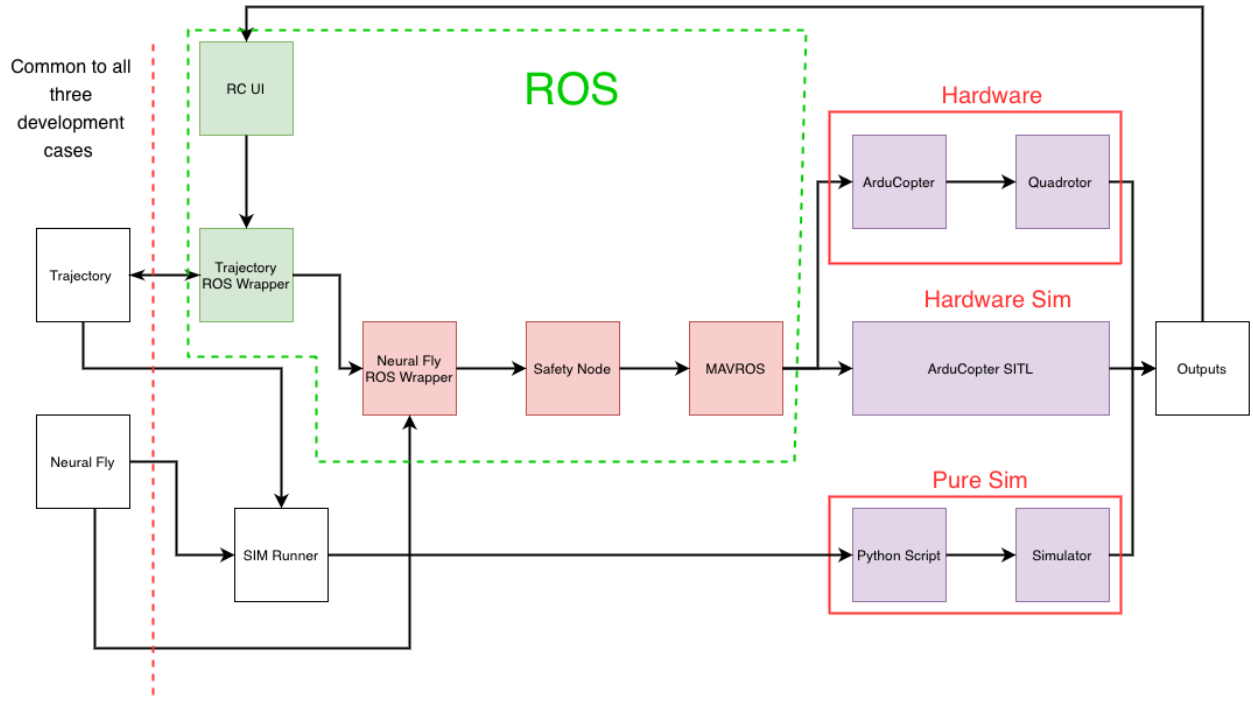

Fig. 1 Neural Fly Code Diagram. I am working on the Neural Fly ROS wrapper so the simulator can interface with the SITL and hardware.

II. Timeline

At the beginning of my SURF fellowship, I created a timeline (Figure 2) to follow over the course of the research period.

I followed my timeline closely; however, I fell behind a little bit in the last few weeks. There were a few issues with

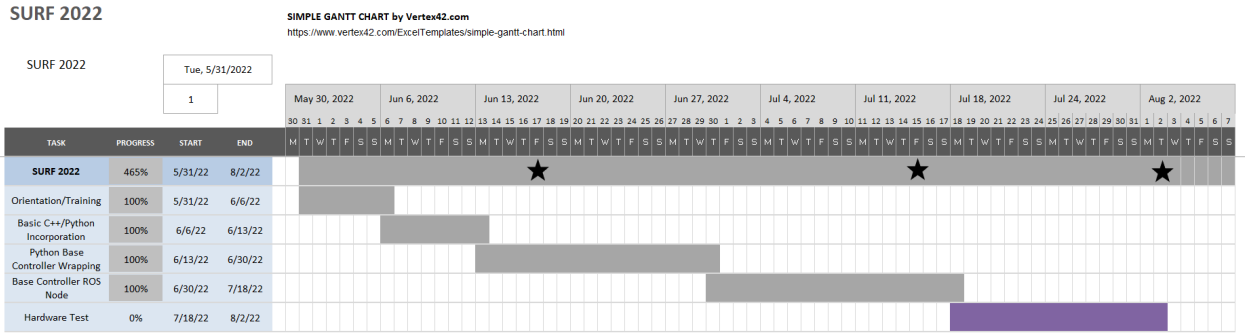

Fig. 2 Gantt Chart depicting the timeline of my SURF internship

creating the ROS interfacing node and properly controlling the quadrotor's attitude. Because of this, I was unable to hardware test until my last week.

III. Methodology

To tackle my project, I decided to start small by incorporating basic Python programs in C++ and build my way up to integrating the simulator controller.

A. Basic Python and C++ Embedding

My goal with this step of my project was to develop a proof of concept that I could integrate Python and C++. I started by writing a simple function in Python that takes in *double* inputs, adds them to variables, then prints out the new variable values. Then, I wrote a C++ program that would grab the variable from Python and echo it back to Python, where it would receive the value, and repeat the process of running it through the function.

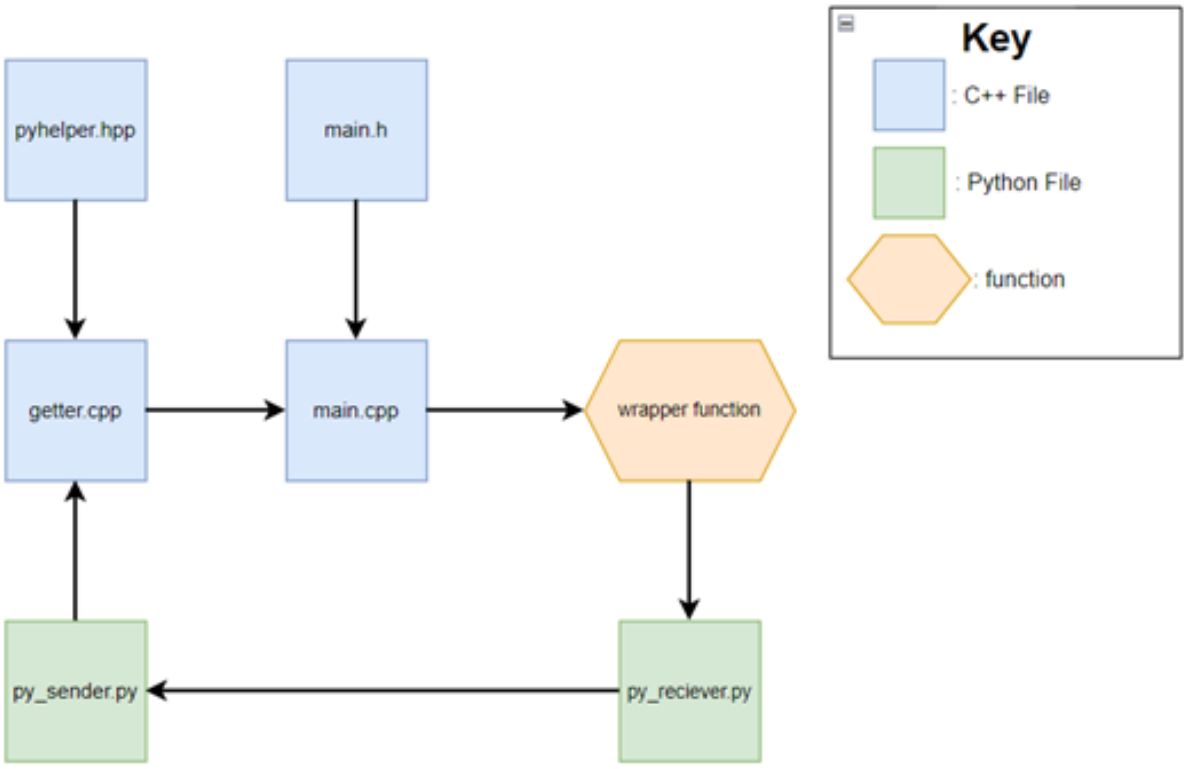

Fig. 3 Code architecture to integrate Python and C++

The biggest challenge I ran into was transforming Python objects into C objects. Unfortunately, a float in Python is not the same thing as a float in C++, so I could not use the same object in both files. Luckily, there were many prebuilt functions in the C library I could use to convert a Python object to a C++ object and vice versa. I ended up not using these prebuilt functions in this step. Instead, I used a function to convert C++ vectors to Python tuples, a function that proved very helpful in the later steps of my project.

Additionally, I ran into a lot of smaller errors that were easy to debug in this step since the code was so simplistic. While fixing the code was easy, it was helpful to run into these errors because I already had an idea of how to resolve them when they would show up again in more complex code.

B. PID Integration

After successfully integrating a simple C++ function, I began to work towards integrating a Python class in C++. This was more difficult than integrating a function because you need to instantiate an object of the class before using any of its functions.

I wrote a simple PID controller class in Python that took a quadrotor's current x-position and used it to calculate the force in the x-direction. Then, I calculated the quadrotor's new x-position based on the force calculation and used a ROS node to send that to the quadrotor. My calculations for the force in the x-directions were flawed, and thus my calculations for the new x-position were incorrect, which lead to some strange behavior from the quadrotor, however, I had proof of concept and a technique for integrating a Python class in C++.

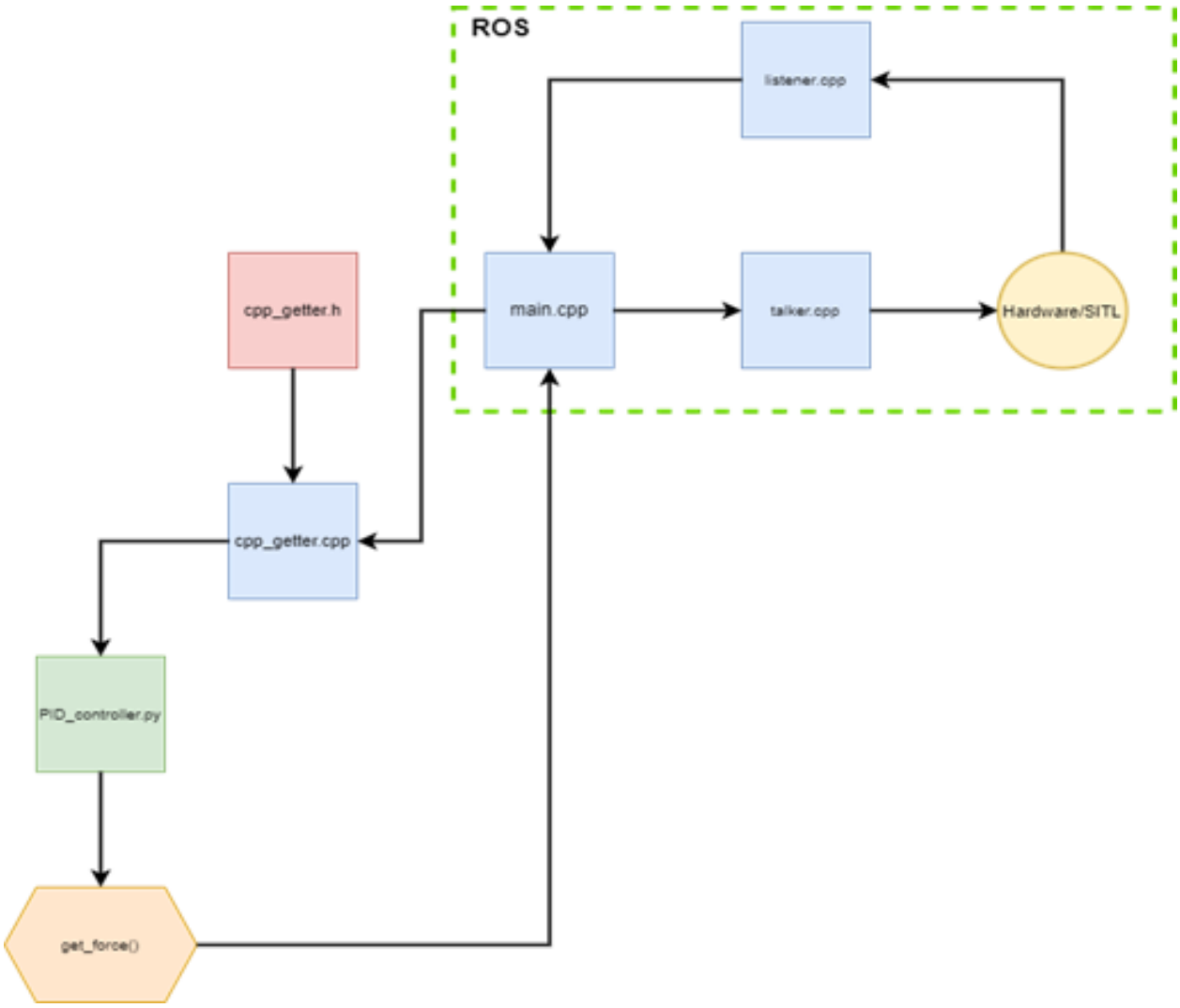

Fig. 4 Integrating a PID Controller to control a quadrotor's x-position.

C. Simulator Controller Integration

After learning how to use a Python class in C++, I started integrating the actual position control from the Python simulator into C++. The simulator position controller took in two parameters—two Named Tuples *, one with a Numpy † Array of the current position, velocity, and acceleration, and another with a Numpy Array of the desired position, velocity, and acceleration. This posed to be a problem because I could not create Numed Tuples and Numpy Arrays in C++, which meant that I had to find an alternative way to pass the data through the controller, instead of just doing everything in C++ and importing anything necessary from Python, as I intended.

I decided the best approach was to do most of my computing in Python instead of C++. This meant that while I was still using C++ to talk and receive data from the quadrotor, I had to then send that data to Python and retrieve the new data from Python. To do this, I took the values we needed from the quadrotor and transformed them into Python floats and put them in Python lists. Then, I sent the values to Python, where they were transformed into Numpy Arrays and placed in Named Tuples. From there it was easy to run the position controller and send the returns to C++ where they could then be sent to the quadrotor through ROS.

*Named Tuples are a type of Tuple that assign meaning to each position in a Tuple so that you can access fields by name instead of by position index. They are part of the collections library.

†Numpy is an external Python library that supports large multidimensional arrays and has a large collection of mathematical functions to operate on the arrays

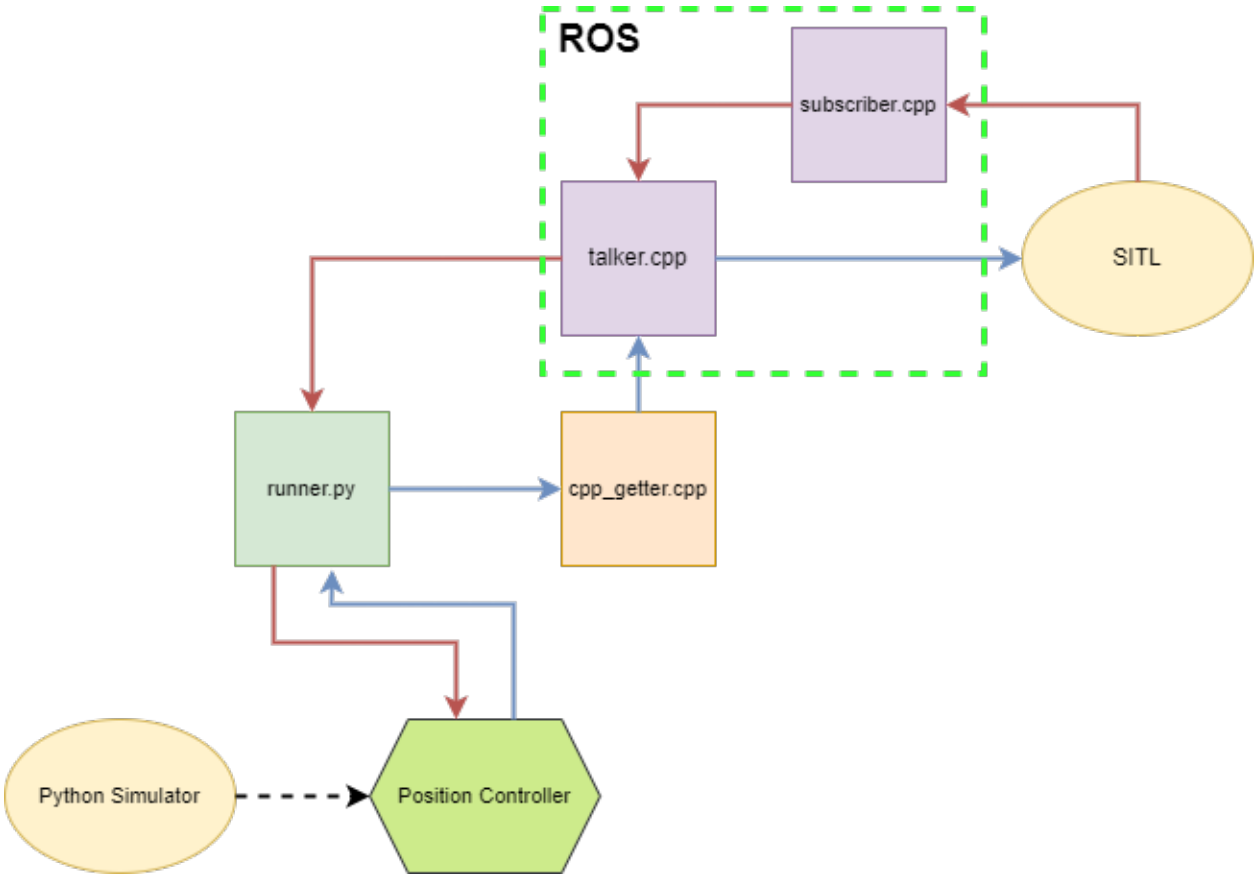

Fig. 5 Controller Integration. The path the data from the quadrotor follows is indicated by red arrows, and the path the returns follows is indicated by blue arrows. The main elements I developed are the non-ROS parts of the talker node, the C++ getter file (cpp_getter.cpp), and the python script, runner.py.

IV. Testing

A. SITL Testing

While integrating the PID controller, and later, the position controller, I tested the ROS nodes using the ArduPilot SITL simulator. ArduPilot is an open-source autopilot system that is used to simulate how code will run on hardware.

I started by testing my PID controller in the ArduPilot SILT. I did this to practice ROS integration and gain a better understanding of how it works so that when I moved on to integrating the position controller, there would not be any gaps in my knowledge. As mentioned above, I was calculating force incorrectly, so instead of flying in a horizontal line, I was flying at an angle. Regardless, I had a better understanding of ROS and how I was going to implement the next step—integrating the position controller in ROS.

Fig. 6 Running the PID Controller on ArduPilot

My next step was integrating the position controller in ROS. Using ROS, I published an attitude target, telling the drone what roll, pitch, yaw, and thrust, and received the quadrotor's current position and velocity, which I could then pass into the position controller.

I ran into a few issues with the ROS integration–my code was not sending or receiving information properly because the channel frequency was zero. This was hard to pinpoint because the issue was within the ArduPilot settings, with which I am very unfamiliar. Once that issue was figured out, I moved on to my other big issue—I was crashing every time I flew in the simulator.

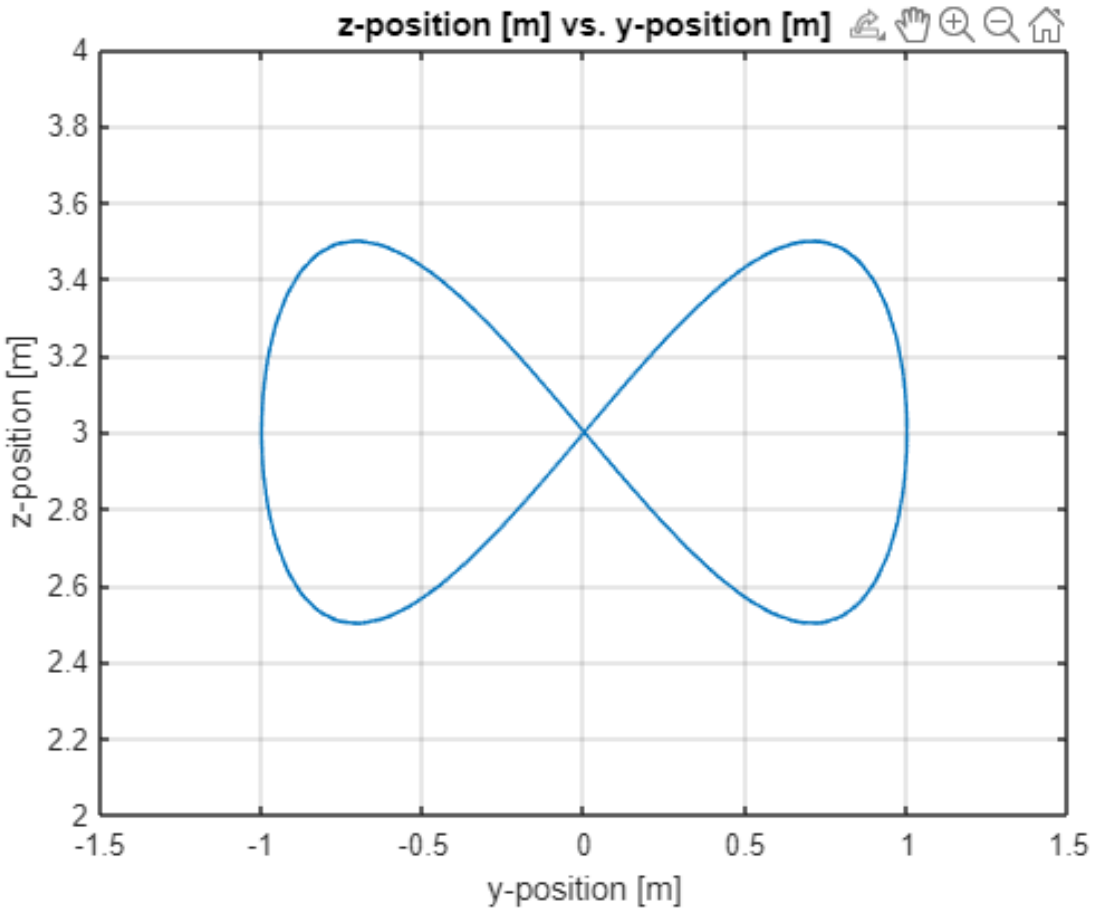

Fig. 7 Planned Flight Path for SITL testing and Hardware testing. I decided to fly in a vertical figure 8 formation to check altitude control. The parametric equation for this plot was: $x(t) = 0, y(t) = sin(t), z(t) = sin(t)cos(t) + 3$

To fix this issue, I began sending a roll, pitch, and yaw, instead of a quaternion attitude target. Additionally, I set limits to how large or small the thrust could be so there was not any rapid ascension or descension. I'm not sure why roll, pitch, and yaw worked when the quaternion didn't because roll, pitch, and yaw were created using the rowan [‡] *to_euler()* function, which changes an input quaternion into Euler angles—if I had more time I would investigate this change and why it works.

B. Hardware Testing

Once I began getting consistent results from the SITL simulations, I began to plan a hardware test outside to see my code work in real life. On Wednesday, July 27, we went to Caltech's North Field and tested the same flight path, I tested in the ArduPilot SITL 7.

During the testing outside, we had one successful test and one failed test. In the successful test, the quadrotor followed a figure-8-like trajectory. While the figure 8s were bad and the altitude of the quadrotor dropped more than I expected, the quadrotor essentially did what I wanted and followed my planned steps.

[‡]Rowan is a Python package for working with quaternions

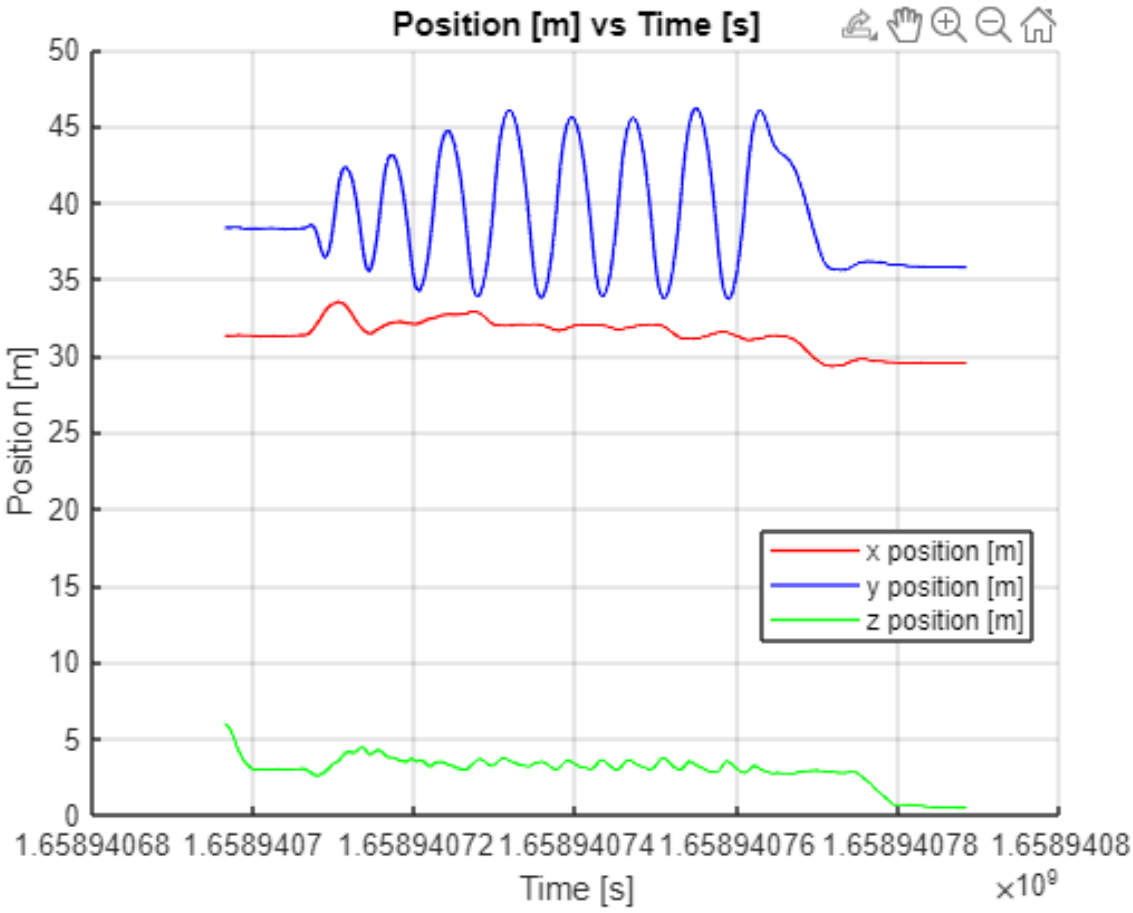

Fig. 8 Position of the quadrotor in the xyz directions [m] vs time [s]. The time is the seconds since unix epoch, January 1, 1970

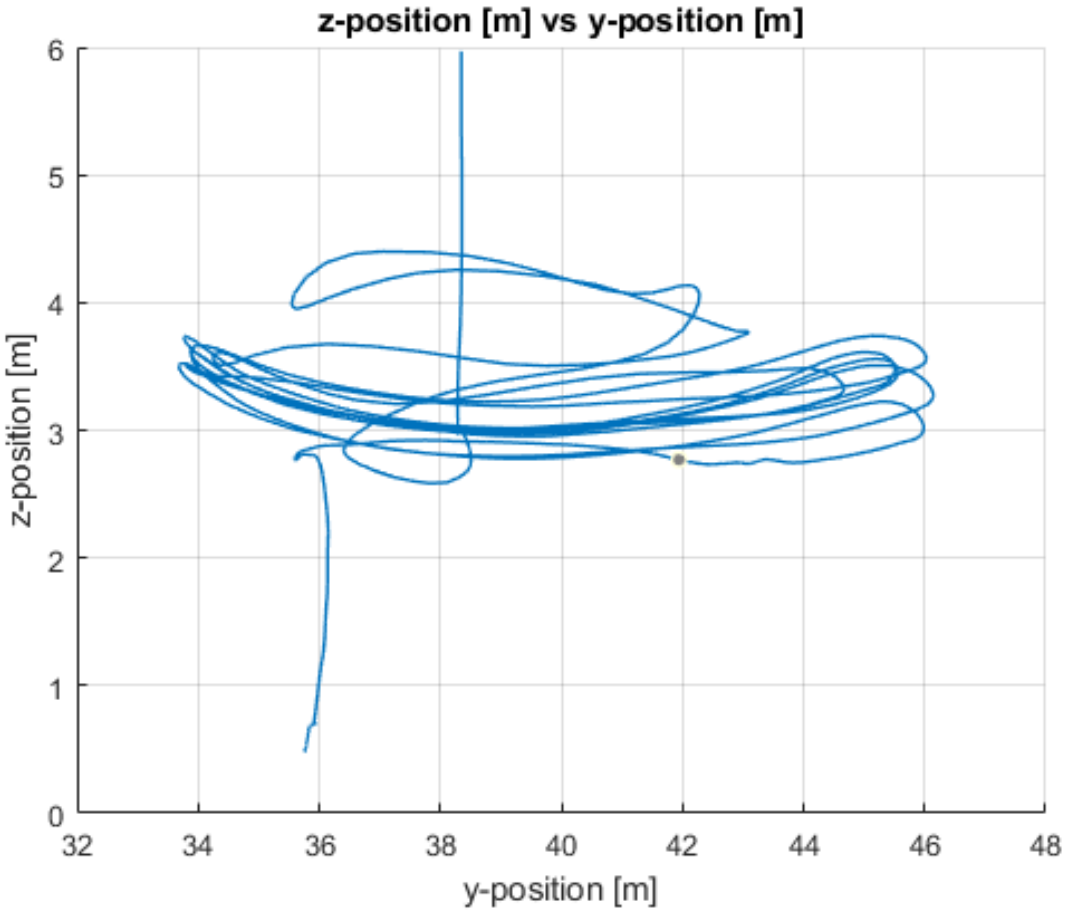

Fig. 9 Position of the quadrotor in the z-direction [m] vs the position of the quadrotor in the y-direction [m]. While the figure 8s are not perfect, the quadrotor is following a path like a figure 8

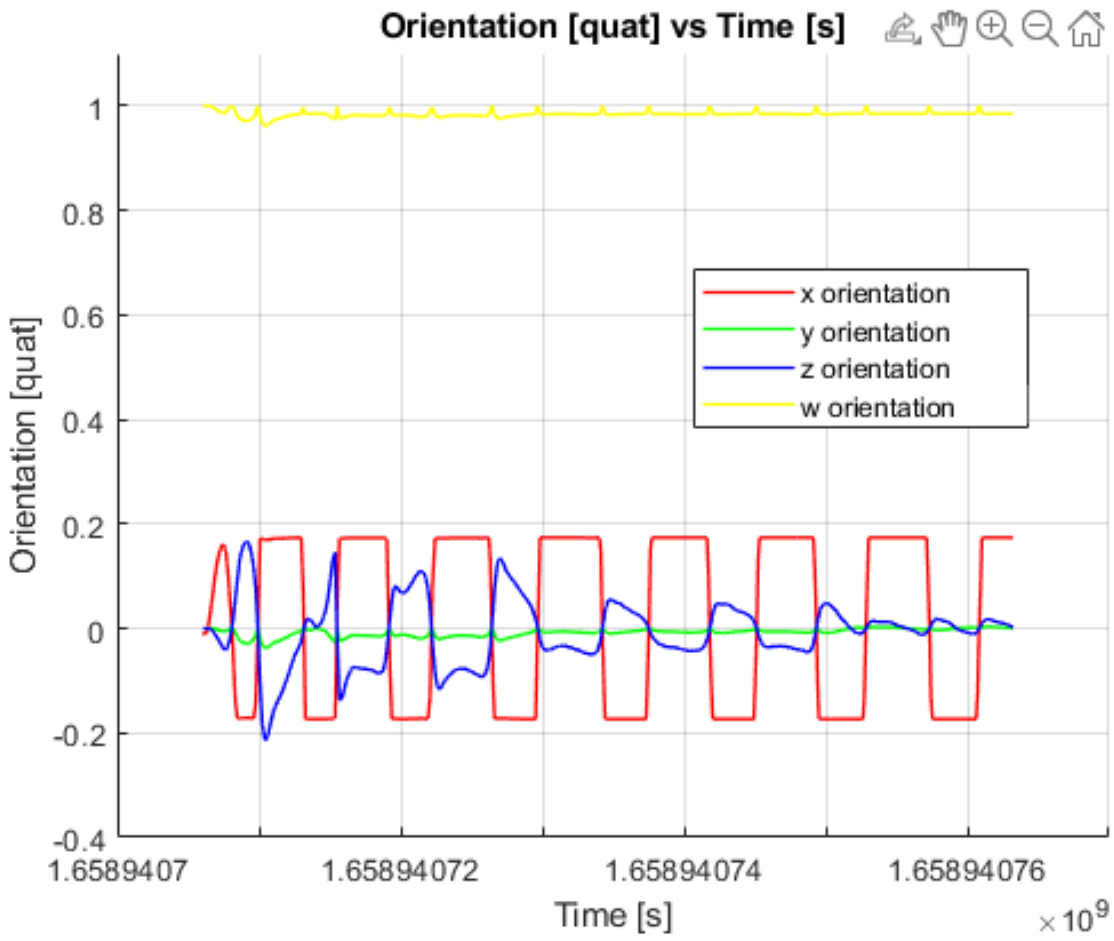

Fig. 10 Orientation of the quadrotor [quat] vs time [s]

After the successful flight, I made a couple of changes to my code, so the quadrotor's altitude was a bit higher (5 m instead of 3 m), and the attitude angles were more aggressive (-30 degrees < roll < 30 degrees). However, when we tried to test the code on the quadrotor, it crashed.

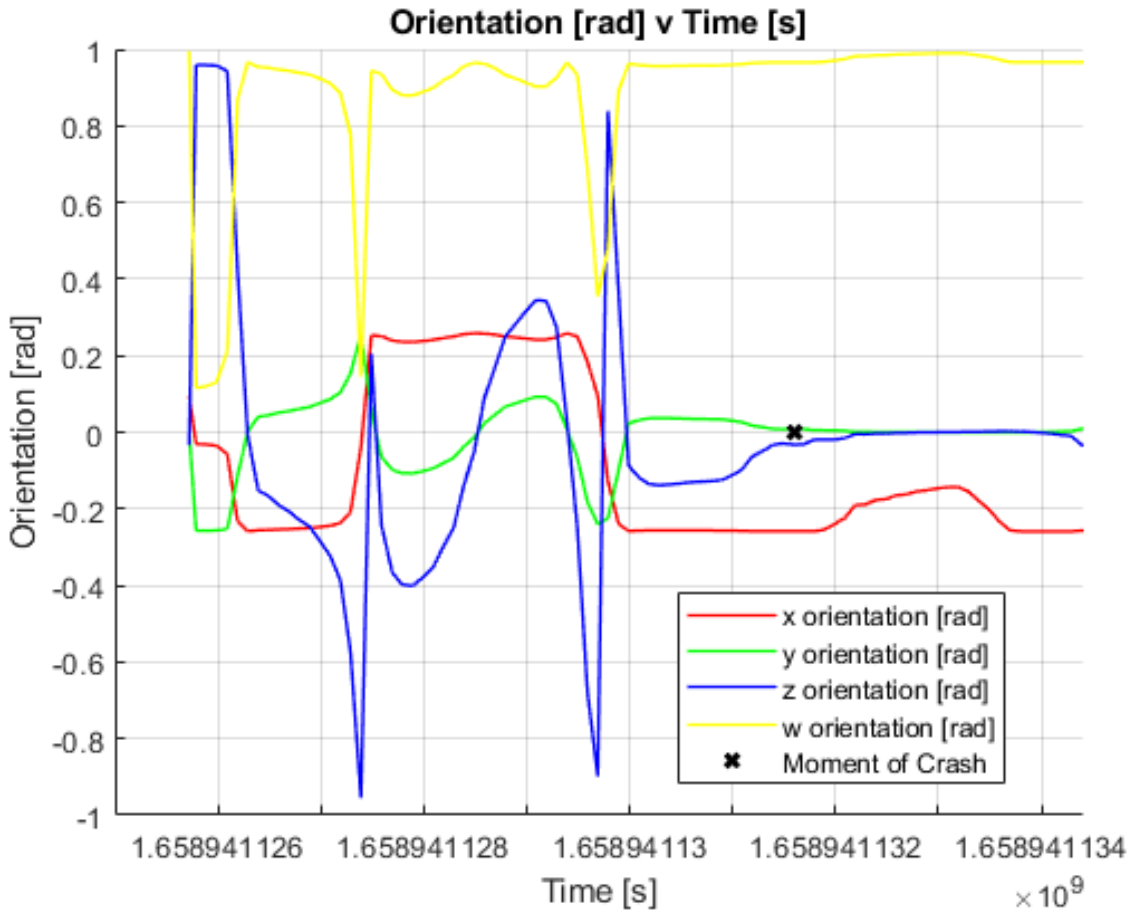

Fig. 11 Position [m] vs time [s]. The time of the crash is marked with an 'x'

Looking at the flight data, I was able to begin to pinpoint why the quadrotor crashed. Right before the crash, both the z-position dropped dramatically. Due to previous issues in stability, I was limiting pitch, however, with the changes in attitude I made earlier, it meant yaw had to become larger to compensate, and thus the quadrotor crashed. Additionally, since the quadrotor was moving at a high velocity, there was no time for the quadrotor to try and catch its fall. After the failed test, I went back to the code and fixed the pitch limitations to be a little higher.

Acknowledgments

I would like to thank Professor. Soon-Jo Chung for letting do research with his lab group. Additionally, I'd like to thank Matt Anderson and Mike O'Connell for working closely with me and mentoring me throughout my fellowship. I'd like to acknowledge the SFP office at Caltech for granting me this opportunity.

References

[1] O'Connell, M., Shi, G., Shi, X., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.-J., "Neural-fly enables rapid learning for agile flight in strong winds - science," , May 2022. URL https://www.science.org/doi/10.1126/scirobotics.abm6597.

[2] "The PID Controller Theory explained," , June 2022. URL https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html.

[3] Collins, D., "Motion basics: How to define roll, pitch, and yaw for linear systems," , May 2020. URL https://www.linearmotiontips.com/motion-basics-how-to-define-roll-pitch-and-yaw-for-linear-systems/.

OUR-J PRESENTS

A Low-Cost, Active Tracking Approach for an Antenna Control Unit (ACU) Ground Station

Protecting Geostationary Satellite Services using the Equivalent Power Flux Density (EPFD) Algorithm

Searching for Correlations Between Lunar Crater Features and Location with Machine Learning

Initial Testing of a 3D Printed Gas Diffuser for Hall Effect Thrusters

Estimating the Probability That the Explosion of an Ink Sphere Produces a Dictionary

Untangling the Neural-Fly Code