Summer 8-2-2021

Power-over-Tether Unmanned Aerial System Leveraged for Trajectory Influenced Atmospheric Sensing

Daniel Rico
University of Nebraska - Lincoln, daniel.rico05@gmail.com

POWER-OVER-TETHER UNMANNED AERIAL SYSTEM LEVERAGED FOR TRAJECTORY INFLUENCED ATMOSPHERIC SENSING

by

Daniel A. Rico

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Carrick Detweiler and Professor Francisco Muñoz-Arriola

Lincoln, Nebraska

August, 2021

POWER-OVER-TETHER UNMANNED AERIAL SYSTEM LEVERAGED FOR TRAJECTORY INFLUENCED ATMOSPHERIC SENSING

Daniel A. Rico, M.S.

University of Nebraska, 2021

Adviser(s): Carrick Detweiler and Francisco Muñoz-Arriola

The use of unmanned aerial systems (UASs) in agriculture has risen in the past decade and is helping to modernize agriculture. UASs collect and elucidate data previously difficult to obtain and are used to help increase agricultural efficiency and production. Typical commercial off-the-shelf (COTS) UASs are limited by small payloads and short flight times. Such limits inhibit their ability to provide abundant data at multiple spatiotemporal scales. In this thesis, we describe the design and construction of the tethered aircraft unmanned system (TAUS), which is a novel power-over-tether UAS configured for long-term, high throughput atmospheric monitoring with an array of sensors embedded along the tether. This was accomplished by leveraging the physical presence of the tether to integrate an array of sensors. With power from the ground station, the TAUS can acquire continuous volumetric data for numerous hours. The system is used to sense atmospheric conditions and temperature gradients across altitudes. We present the development of the prototype system, along with a discussion of the results from field experiments. We discuss the influence that power losses across the tether have on the sensors' abilities to accurately sense atmospheric temperature. We demonstrate a 6-hour continuous flight at an altitude of 50 feet, and a 1-hour flight at sunset to acquire the gradually decreasing atmospheric temperature from an array of 6 sensors. We then modeled the TAUS and sensor

array to computer simulate four trajectories (mower, spiral, star, and flower) for the TAUS and evaluated the system and sensing performance via well-defined factors. We conducted outdoor experiments to characterize system performance while in operation and to inform the development of models and trajectory simulations. From the analysis of the experimental data, we found minimal sensing error with respect to ground truth installations at comparable altitudes. Leveraging the simulated trajectory outcomes we reconstructed the changing input temperature fields. The analysis of the simulated data indicated that the power-tethered Star trajectory performed well with respect to key performance factors when measuring changing atmospheric fields. The TAUS will be improved by incorporating multi-variable sensors and an optimal control algorithm for elevated levels of operational autonomy.

DEDICATION

To a better tomorrow. This work serves as a means to a very important end.

To my friends and family, without whom I would not be who I am today.

To my laboratory and the expectations set by those who have come before me.

To my advisors, Dr. Carrick Detweiler and Dr. Francisco Muñoz-Arriola, for their empathy, patience, and continuous support in mentoring me to be a better engineer and scientist. No graduate student could have ever asked for better guidance. For a period during this endeavor, it was the darkest of times for me personally, which reduced me to nothing short of a liability. Neither of them gave up on me when others would have. The work presented here was only possible because of them.

ACKNOWLEDGMENTS

We would like to thank the University of Nebraska-Lincoln, Department of Computer Science & Engineering, Department of Biological Systems Engineering, School of Natural Resources, and members of the Nebraska Intelligent MoBile Unmanned Systems (NIMBUS) Laboratory. Special thanks to Dr. Justin Bradley for being the third committee member and providing feedback on our work. Special thanks to Dr. Ajay Shankar for his assistance with matters both theoretical and practical throughout this work. We also thank the facility managers Ms. Jenny Stebbing and Mr. Stuart Hoff of the UNL-Havelock research facility and UNL-Rogers Memorial Farm research facility, respectively.

GRANT INFORMATION

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), the United States Department of Agriculture (USDA), or the Robert B. Daugherty Water for Food Global Institute (DWFI). This work was supported in part by NSF-DGE-1735362, NSF-IIS-1925052, NSF-IIS-1925368, NASA-ULI-80NSSC20M0162, USDA-NEB-21-166-1009760, USDA-NEB-21-176-1015252, and the DWFI Student Support fund.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the years, the agricultural sector has continuously used various soil and meteorological sensors, and robotic systems in their work [1] [2]. Successful and sustained applications have taken the form of automated field machinery [3] [4], weather stations [5], and a network of soil moisture sensors for precision irrigation [6], among many other applications. These systems have helped increase efficiency and production in the agricultural sector.

In recent years, unmanned aerial systems (UASs) have become more ubiquitous and offer the ability to acquire data from various altitudes and resolution that was traditionally hard to access. UASs have found their way into agriculture with applications in environmental and phenotype monitoring via red, green, blue – depth (RGB-D) [7], normalized difference vegetation index/near-infrared (NDVI/NIR) [8], multi/hyperspectral (HS) imaging [9] [10] [11], and precision agriculture through the integration of various Machine Learning and Internet of Things (IoT) concepts [12]. Non-imaging agricultural application have taken the form of wireless sensor networks [13], atmospheric sensing [14], crop height estimation [15], and precise applications of pesticide [16], among other proximal sensing capabilities. Typically, UASs are assumed to be a multi-rotor variant, but even fixed-wing UASs have shown promise in providing the sector data [17]. Data

from onboard imaging and environmental sensing are more pertinent, unique, and readily available to farmers, improving our abilities to predict phenotypes.

Despite the advantages UASs provide, they have practical limitations. Traditional UASs and current commercial off-the-shelf (COTS) systems have limited payload, constrained flight times, and can only collect data at one point in space at a given instant. However, Lussier *et al.* [18] show that integrating photovoltaic or hydrogen fuel cells on-board a DJI Matrice-600 could theoretically increase *hover* flight times from $\sim$32 minutes to $\sim$ 52.5 minutes (+164%). Regardless, the addition of photovoltaics and hydrogen fuel cells are expensive and favor larger more payload capable UAS. Additionally, COTS systems overall lack autonomous operation, an aid in user-friendliness and system integration. A flight duration of $\sim$30 minutes round-trip from a higher-end COTS system is limited in representing both space and time, leading to constrained agricultural decisions, predictive evaluations, and applications. Based on a survey of agricultural decision-makers [19], particularly producers, atmospheric temperature data ranked a close second to soil moisture amongst the most important information. Temperature data aids in decisions about planting, harvesting, defoliating, crop modeling, and preventing diseases and pests, among others. To get this data agricultural producers depend on sparse meteorological stations/towers, remote sensing, and rarely vertical profiling UAS to acquire low-spatiotemporal resolution data to drive large-scale weather and climate models which will produce extrapolated outcomes. Atmospheric scientists and agricultural practitioners often use meteorological stations, radiosonde, and eddy covariance (EC) towers to sample the lower troposphere. This provides valuable information about its interdependence with the land surface to improve process understanding, weather forecasting, and the design and operation of infrastructure. These forms of technology currently acquire a spatially disjoint

and discrete-time series representation of the Earth's atmospheric profile. UASs have eased this acquisition at frequencies and resolutions that were traditionally difficult to sample while dramatically increasing the density of measurements. For atmospheric sensing via UASs, an array of atmospheric sensors could generate high-spatiotemporal resolution data with elevated levels of continuous throughput. Thus, there exists a gap in knowledge and an agricultural/atmospheric science application for a UAS that can autonomously deploy in isolation for long-term self-sustain operation at altitude to acquire continuous high-spatiotemporal resolution atmospheric temperature data. Furthermore, as a basic and easily sensed physical variable, the atmospheric temperature can be used to assess the robustness, reliability, and suitability of the platform.

This leads us to the following set of Research Question(s):

1. **Can we develop a proof-of-concept UAS that can increase flight times and data throughput (without profiling) for the agricultural and atmospheric sectors?** We *hypothesize* that we will be able to accomplish this if we engineer a UAS powered over a tether that leverages the physical presence of the tether for placing meteorological sensors. This will be accomplished by engineering a containment structure to house the entire system which will include a battery bank capable of sourcing enough electrical power to keep the system operational for multiple continuous hours, transmission power electronics, sensor/control/communication circuitry, mechatronics related to the autonomous operation and tether management, and a COTS UAS retrofit with step-down power electronic and quality positioning hardware. In doing this, we predict that the system will gain increased flight times and throughput. The development of a hardened system designed to sample

the atmosphere for nearly indefinite periods in the wild is a ***challenging*** engineering problem. The system will need to be robust and hardened for independent operation in remote agricultural locations for active deployment. Chapter 3 answers this question and expands on details surrounding the topic.

2. **Will the power transmission efficiency bias the atmospheric temperature sensors?** We *hypothesize* that if the power transmission efficiency is below a certain unknown percent the power-tether conductors will experience significant electron-based quantum collisions and that will result in excessive heating which will radiate through the proximal atmosphere biasing the sensors. To test this we will need to conduct a field experiment with two sets of sensors with one set positioned on-tether and another set positioned off-tether to run an intercomparison error analysis. Chapter 4 answers this question and expands on details surrounding the topic.

3. **Can we validate the systems' base-level operational performance in the field?** We *hypothesize* that if we take the system to the field to conduct a quasi-static sampling experiment in proximity to a well maintained and quality ground-truth field tower installation we will be able to run an intercomparison analysis to determine sampling error between the two at altitude. To test this we will travel to Rogers Memorial Farm (RMF) east of Lincoln, Nebraska to run the experiment near the 10 meters Above Ground Level (AGL) mesoscale network (MESONET) meteorological tower with atmospheric temperature sensors integrated at 2 meters and 10 meters AGL. Chapter 4 answers this question and expands on details surrounding the topic.

4. **Since the UAS will be tethered and therefore bounded creating a new system limitation, can we evaluate a set of system trajectories against well-defined factors to determine operational performance?** We *hypothesize* that to evaluate the systems theoretical operational performance we will need to first computer model and simulate the system in a physics emulated environment. Since atmospheric temperature is inherently a dynamic spatiotemporal variable we will need to evaluate trajectories of a tethered system that traverses a bounded volume of the atmosphere. We will then need to simulate a tethered system traversing through a temperature field, which can be modeled as a dynamic cold-front tracking into the bounded volume while the system traverses. The data acquired in-flight can be relayed in real-time for analysis and later for post-processing. The trajectories can then be characterized through the simulations. The construction of a computer model for simulations related to trajectories and behavior based on the physics of the actual system poses a ***challenging*** computer science problem. The trajectories need to be well defined and transformed from purely mathematical expressions to software scripts that operated within the bounds of the simulated environment. We predict coupling the trajectory information with a dynamic temperature field will produce a demanding simulation workload and a challenging post-processing environment. Chapter 5 answers this question and expands on details surrounding the topic.

In this thesis, we present our designs of the tethered aircraft unmanned system (TAUS) [20], shown in Figure 1.1. The (TAUS) is a novel power-over-tether-based UAS configured for long-term stand-alone high-throughput environmental monitoring. The TAUS is composed of a direct current (DC) to DC-based power-

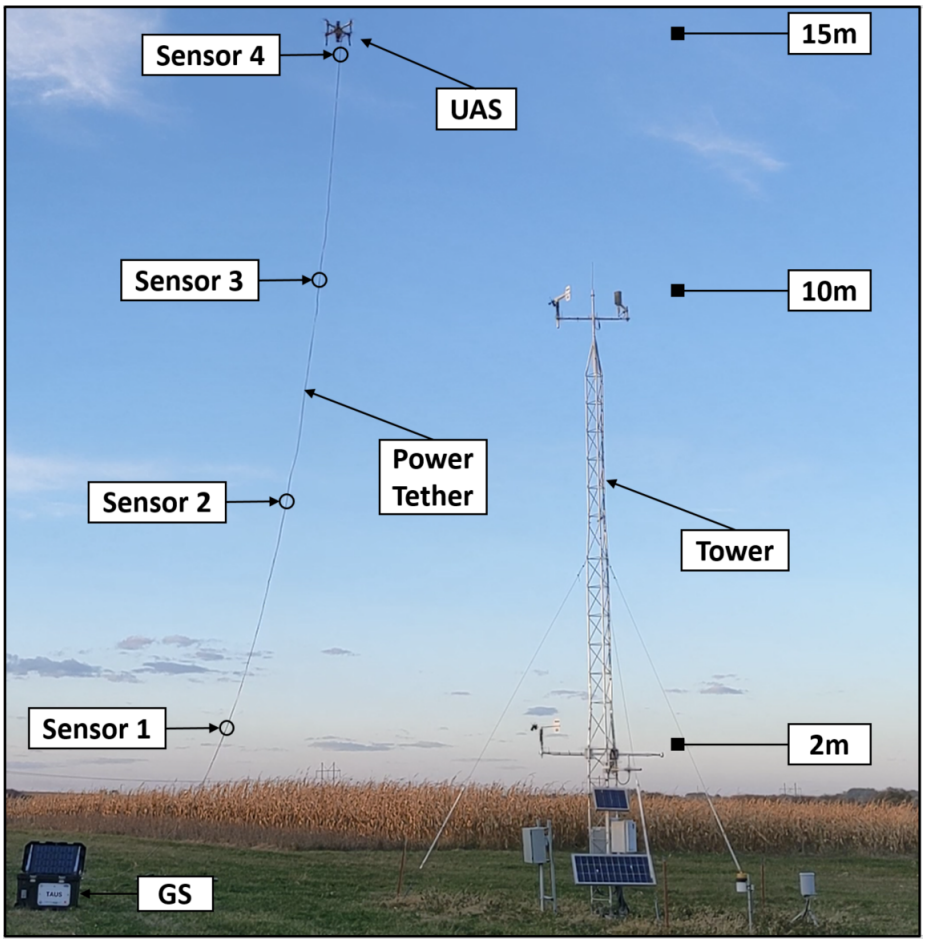


Figure 1.1: The TAUS-Beta prototype in operation alongside a 10m above ground level meteorological tower at Rogers Memorial Farm in Lincoln, Nebraska.

over-tether and an array of discrete temperature sensors. A series of field tests to acquire data for evaluation is presented. The operational performance of the system's sensor array was validated via field experimentation. TAUS provides results comparable with other sampling devices, such as Eddy Co-variance (EC) towers and tethered weather balloons, but with the mobility to sense a volume at a significantly higher spatiotemporal resolution. We also characterize the TAUS trajectories to identify parameters that can be used on the physical system to improve sensing atmospheric variables in three dimensions. Specifically, we evaluated four distinct and well-defined trajectories that detect spatiotemporal

changes in the atmosphere. We defined multiple factors such as volume sensed, percent coverage, and temperature field reconstruction error to objectively quantify system factors for performance evaluation. This allowed us to determine the viability and impact of selecting a particular trajectory.

1.1 Contributions

Part of this thesis related to the development and agricultural-atmospheric application of the TAUS is published as a proceedings paper to the American Society of Agricultural and Biological Engineers (ASABE) Annual International Meeting (AIM) 2020 [20]. Part of the thesis related to the trajectories, simulations, and evaluations of the theoretical performance of the TAUS was accepted for publication on June 30th, 2021 as a proceedings paper to the Institute of Electrical and Electronics Engineers (IEEE) international conference on Intelligent RObots and Systems (IROS) 2021 [21] In summary, the main contributions of this work are:

1. Development of a novel power-over-tether UAS prototype that includes sensors placed along the tether to provide high spatiotemporal resolution sampling of atmospheric temperature.

2. Field experiments and results demonstrating validation of the operational performance of this platform to sense temperature fluctuations within a 15m above ground level range.

3. Development of a model that represents the physics governing power-over-tether UASs in flight.

4. Design and implementation of power-over-tether trajectories for the model to evaluate theoretical performance via factors in a bounded volume of a

simulated atmosphere.

This thesis is organized as follows: Chapter 2 covers the related published literature along with industry applications of similar technology, Chapter 3 describes the development of the system from Alpha to Beta coupled with field experimentation to determine operational performance, Chapter 4 covers model development and trajectory simulations through static and dynamic temperature fields, and a discussion of the work along with the conclusions are presented in Chapter 5.

Chapter 2

Related Work

Conventional UASs have always had a limitation of flight time due to constrained payload and battery capacities. Thus, other academic and industrial researchers have previously developed power-over-tether UAS geared toward long-term flight applications. However, these systems have not leveraged the power-tether for anything other than power transmission, such as atmospheric sampling. To claim a novel and practical contribution to atmospheric sampling technology, we differentiate our work with the current state-of-the-art technology spanning conventional and unconventional forms. Similarly, we present examples of untethered and power-tethered UAS models and simulations to demonstrate the knowledge gap that exists for modeling novel power-over-tether UASs configured for atmospheric sampling by incorporating atmospheric sensors along the physical tether.

Prior work on the development of power-over-tether UAS technology is discussed in section 2.1, conventional atmospheric sampling technology is discussed in section 2.2, applications of UAS with respect to sampling the atmosphere is discussed in section 2.3, and both conventional and tethered UAS modeling and simulations are discussed in section 2.4.

2.1 Power-over-Tether UAS Technology

Power-over-tether UASs have previously been developed [22] with applications that include collaborative navigation [23], enhanced perspectives for teleoperated construction machine [24], and detection of oil pollution from ships [25]. Gu *et al*. [26] sought to develop two unique forms of Tethered Aerial Robots (TARs, a.k.a. power-over-tether) for the application of endurance missions related to nuclear power plants. Specifically, missions to traverse compromised stations and monitor leaked radiation at altitudes for long periods. They developed prototypes of the Roaming-TAR (RTAR) and Stationary-TAR (STAR) systems. In doing this they explored the fundamental question of which power transmission technique to employ in their work concerning alternating current (AC) or direct current (DC) power electronics. There is two main disadvantage to using AC for aerial applications: (1) inverter, transformers, filters, and resonance circuitry, and (2) the weight related to the components/circuits. A UAS payload is at a premium and power electronic hardware will be impactful. They experimentally verified their electrical engineering theory which concluded the DC-based STAR performed the best concerning power efficiency and payload weight minimization. Their work was influential in the decision to make the TAUS a DC system.

With respect to industry, there are numerous companies founded in recent years that leverage this technology to accomplish various tasks such as video surveillance [27] [28], military field-deployable aerial reconnaissance [29], and even cleaning off high and hard-to-reach structures with pumped water and solution [30].

A number of the aforementioned systems are presented in Table 2.1 for system-to-system comparison. The power-tether in all of these cases is used exclusively

for power transmission and has not been leveraged for other applications.

Table 2.1: Previously developed research prototypes and commercially available power-over-tether systems with their system specifications listed if available. The table is divided into research and commercial systems, respectively.

System	T-Length [m]	T-Mass [g/m]	Power [W]	Sys-Mass [g]
Gu: RTAR-AC	20	2	183	580*
Gu: STAR-AC	80	2	306	238*
Gu: STAR-DC	80	2	308	124*
Hoverfly‡: -	-	-	-	-
Elistair: LIGH-T-v4	70	10.5	2500	20,000†
Elistair: SAFE-T2	130	25	2800	25,000†
Powerline‡: -	-	-	-	-

Note: Only *aerial platform and tether, and †ground station and tether, respectively.
‡These companies did not respond to system inquiries when writing this thesis.

One of the major outcomes of power-over-tether UAS technology is the ability to conduct endurance missions significantly longer when compared to conventional onboard battery-based flight. This ability has previously been achieved with not just power-over-tether, but with concentrated electromagnetic waves. In 1964 members of Raytheon Spencer Laboratory demonstrated the proof-of-concept for the first microwave-powered heavier-than-air helicopter flight with a string-type rectenna sourced microwave beam [31]. Achtelik *et al*. [32] leveraged light amplification by stimulated emission of radiation (LASER), otherwise known as LASER power beaming. They were able to fly a small UAS continuously for $\sim$12 hours which broke the 2011 record for longest quad-rotor flight. None of the aforementioned examples of power-over-tether systems have leveraged the power-tether for anything beyond power transmission.

We differentiate ourselves from the aforementioned examples by incorporating an array of atmospheric sensors along our power-tether so we can simultaneously sample across the vertical profile of the troposphere for extended periods. By doing

this we not only increase the system operational time but its sampling throughput by a factor of the number of sensors deployed along the power-tether. To our knowledge, our project is the first attempting to leverage the physical presence of a power-tether to acquisition a gradient of atmospheric data, in particular, surface atmospheric temperature. Additionally, our system is engineered to operate as a purely DC system isolated from grid AC power infrastructure and it does not require external natural gas or diesel-electric generator supply to operate, which is a novel system characteristic relative to those previously mentioned.

2.2 Conventional Atmospheric Sampling Technology

Current technology that attempts to sample the troposphere is lighter-than-air (LTA) radiosonde (weather balloons) are employed at scale across the globe. For instance, twice a day in the conterminous United States, 69 weather balloons are simultaneously released to acquire temperature, pressure, and relative humidity as the balloons ascend to approximately 35 km, relaying sensed data every 1-2 seconds for approximately 2 hours. This low-resolution spatial and temporal data is then used to drive weather models to help make weather predictions even locally despite major generalizations due to the low spatial and temporal resolution. This form of technology is often deployed without a physical tether and, therefore, not recovered due to a lack of system position control and significant physical lateral drift up to 300 km caused by wind [33]. Laroche *et al*. [34] found that the potential radiosonde drift due to the influence of wind has been shown to harm high-resolution data accuracy and short-range forecasts in the upper troposphere and stratosphere. Therefore, they developed a method for estimating missing balloon drift position data points from radiosonde spatiotemporal data.

By incorporating this estimation for radiosonde drift they were able to improve medium-range forecasts as well. When radiosondes are tethered (tethersonde) they can offer similar sensor distribution advantages but the cable has been used for physical tethering and sensor placement and not as a means of power transmission. Additionally, the tethersonde and especially the radiosonde have no wind compensation or fine/coarse control over lateral or vertical positioning. The upward buoyant force produced from the Helium in the balloon may also have to combat a downward wind force vector component making steady tethersonde altitudes also challenging and uncertain. Unlike these conventional forms of atmospheric sensing, we introduce the means to deploy and hold a position in space while simultaneously sampling.

Another current technology that samples the atmosphere at constrained spatial extents is the aforementioned EC towers. This technology samples the troposphere's vertical profile at a high frequency from ground level to fixed heights typically ranging anywhere between 8m and 75m. The sampled data is typically used to derive air quality via concentrations of gases and their flux, as well as gas and water exchanges between the atmosphere and the land surface. Tower installation is relatively expensive, and there are only about 800 active and registered towers with the FLUXNET global network database [35], and 300 active and registered towers with the AmeriFlux network database [36]. This means there is low spatial resolution, and since these are physical towers intended for long-term sensing, this technology is not readily deployable.

Unlike radio and tethersonde technology, our system is capable of maintaining a relatively fixed position in space and it is designed to return to the launch position for rapid re-deployment. This fixed position behavior emulates conventional meteorological towers but with the added benefits of user-defined altitude, ground

deployment position, and significantly lower costs.

2.3 Atmospheric Sampling with Kites and UAS

Kites are one of the oldest and well-established means of atmospheric exploration and sampling for humanity going back at least 250 years. There are many instances of COTS and novel research kites having success in the troposphere with sensor suites tracking in the vertical with Tether Lifting Systems (TLS) [37] such as WindTRAM [38] and the Sliding Weather Instrument Fixed to Tether (SWIFT) [39]. Multi-spectral imaging for mapping landscapes has even been accomplished using kite technology [40]. Irvin *et al.* [41] developed a novel kite system called the High Altitude Weather Kite (HAWK) [42] that incorporate lightweight inflatable structures. They have deployed to an altitude of 2,755m above ground level (AGL) allowing for sampling across the atmospheric boundary layer (ABL). Similar to other conventional forms of atmospheric sampling, kites do not have significant ability to fix themselves in space or carry out trajectories limiting their dynamic contribution. Additionally, kites tend to need windy conditions to operate which is a deployment limitation.

There are numerous instances of conventional UASs retrofit with sensors integrated on-board used to sample variables such as aerosol concentrations [43], and wind vectors [44], among others. An attempt to use a typical battery-powered hexacopter UAS to sample the troposphere to address the drift limitations of weather balloons were carried out with success [45], but on-board sensor performance was not rigorously evaluated against quality sources of ground-truth in the field. Additionally, this convention limited atmospheric data acquisition to only the ascent and disregarding the descent due to propeller down-wash and

disturbance. Islam *et al.* [46] developed a housing and physical configuration of on-board sensors that sample the troposphere in both ascent and descent for atmospheric variables such as pressure, relative humidity, and temperature (PHT). When profiling, they have essentially doubled system sampling throughput with their housing. This is because decent data is routinely discarded due to multi-rotor generated down-wash disturbing the air below the system. A degree of optimal was argued due to their rigorous testing and cross-validation through inter-comparison against other retrofit UAS and quality undisturbed sources of ground-truth installations in the field. Less has been done concerning atmospheric sensing with power tethered systems. UASs have been used to suspend a distributed temperature sensing (DTS) fiber optic non-powered tether to acquire short but high-resolution profile measurements of the atmospheric boundary layer [47].

Kite technology is highly dependent on adequate wind velocity to deploy, whereas our system is not dependent on outside forces for aerodynamic lift. This significantly increases our systems deployment opportunity and independence. A typical UAS retrofit with an atmospheric sensor can only sample at one position in space at any given instance. Unlike these approaches, we have integrated an array of spatially independent sensors, which therefore increases meaningful spatiotemporal sensing throughput by a factor of the number of deployed sensors. This provides continuous sampling of the lower-atmospheric column at vertical increments designated by the sensors along the power-tether. Unlike the UAS coupled with the optical DTS system, we are power-tethered which significantly increases our flight time. Also, our atmospheric sensors can be upgraded to multi-variable sensors to not only increase throughput but sampling diversity whereas the DTS system is fixed to atmospheric temperature.

2.4 UAS Modeling and Simulations

Typical multi-rotor UAS models have been successfully simulated in programmable physics engine environments based on the Robot Operating System (ROS), such as GAZEBO®. UAS applications for this type of simulator have included hardware-in-the-loop (HWL) [48] and machine vision [49], among others. Another ubiquitous simulator is MATLAB Simulink®, which uses its physics engine. Simulink has been used to develop the means to validate critical UAS software [50], among other applications.

Purely mathematical simulations are widespread. Three-dimensional path generation and tracking for a UAS were simulated with this approach [51]. Concerning power-over-tether-based UASs, this approach has been applied with success on tether catenary shape analysis [52], geometric control [53], among others. We have combined these two approaches by taking the mathematics that describes the UASs motion and integrating it with a UAS-tether model in a physics emulated environment. We have produced simulations of trajectories that will eventually be validated in the field with a physical system.

Houston *et al.* [54] developed simulations of rotary-wing aircraft operating as profilers in a Convective Boundary Layer (CBL) alongside fixed-wing aircraft operated through transects across a simulated air mass boundary. They modeled temperature sensors as a first-order linear time-invariant (LTI) differential equation and found that instantaneous errors scale directly with sensor response time and airspeed for all experiments.

UAS trajectories for coverage path planning (CPP) have been extensively studied [55]. The unmanned aerial vehicle routing and trajectory optimization problem (UAVRTOP) has been well-defined [56]. We have leveraged knowledge related to

UAS CPP, but the UAVRTOP does not account for variables and functionality of a power-over-tether UAS leveraged sensor array.

In each of these respective areas, power-over-tether UASs have either not been considered or they do not incorporate temperature sensor models along the modeled power-tether. Two-dimensional temperature fields have been simulated for static un-tethered UAS to sample with on-board modeled temperature sensors, but power-tethered models have not been developed for these atmospheric sampling simulations. Conventional UAS models have been simulated with numerous unbounded trajectories, but bounded power-tether models have not. We differentiate ourselves by simulating a 3-dimensional power-over-tether UAS and temperature sensing array model traversing a bounded volume with a dynamic temperature field. The influence various trajectories have on theoretical system performance is evaluated against well-defined factors.

Chapter 3

System Development & Evaluation

This chapter aims to answer Research Question 1 stated in Chapter 1: "**Can we develop a proof-of-concept UAS that can increase flight times and data throughput (without profiling) for the agricultural and atmospheric sectors?.**" based on our *hypothesis*: "We will be able to accomplish this if we engineer a UAS powered over a tether that leverages the physical presence of the tether for placing meteorological sensors."

This chapter is organized as follows: Section 3.1 describes the targeted design requirements for our power-over-tether system, and the development of the TAUS alpha and beta prototypes in Section 3.2 and 3.3, respectively. The description of the beta sub-system composition, sensor calibration, and beta retrofitting is in Section 3.3.1, 3.3.2, and 3.3.3 respectively.

3.1 Design Requirements

When engineering systems, it is good design practice to start by well defining key system parameters and overall requirements. Our goal was to develop a proof-of-concept system to answer Research Question 1, which could also be improved through an iterative process of revision. We first decided we wanted our system to

be able to reach a minimum of 15m AGL for field intercomparison experiments for operational validation. We chose this height AGL because common local and standard mesoscale network (MESONET) towers [57] are typically 10m AGL with quality well-kept atmospheric sensors at both 2m and 10m. These installations provide multiple undisturbed data points in the field that will serve as ground-truth to compare our system against at altitude. With this requirement identified, we then lab-tested a COTS quadrotor UAS to determine peak/continuous power demands, and maximum dynamic payload. We then wanted enough power supply for the system to fly for 6 hours continuously as a significant (18-24 times typical flight endurance with standard LiPo batteries onboard) temporal milestone. Based on the published literature discussed in Chapter 2, specifically section 2.1, we decided to make our system entirely based on DC end-to-end. Using DC was found to be the most energy and payload-efficient configuration for power-over-tether UAS technology. This would mean we could leverage the simple DC power relationships 3.1 to calculate nominal values for our system, where P_{DC}, V, I, R, and P_{LOSS} stands for DC Power, Voltage, Current, Resistance, and DC Power Loss, respectively.

$$P_{DC}\left[\frac{Joule}{Second}\right] = V\left[\frac{Joule}{Coulomb}\right] * I\left[\frac{Coulomb}{Second}\right]$$

$$V\left[\frac{Joule}{Coulomb}\right] = I\left[\frac{Coulomb}{Second}\right] * R\left[\frac{Joule * Second}{Coulomb^2}\right] \tag{3.1}$$

$$P_{LOSS}\left[\frac{Joule}{Second}\right] = I^2\left[\frac{Coulomb^2}{Second^2}\right] * R\left[\frac{Joule * Second}{Coulomb^2}\right]$$

Since the length of the tether, the gauge of the conductors, and the number

of conductors are all directly related to the overall tether payload for the UAS we wanted to minimize the number of conductors and maximize the gauge. Concerning the temperature sensors that would be integrated in-line, we selected sensors that would share a single data line, otherwise known as an open-drain circuit configuration. This would mean the sensor array would interface with a microprocessor via a form of inter-integrated circuit (I^2C) communication protocol. A time-varying sinusoidal power such as AC can increase the Bit Error Rate (BER) on data lines that are close to the power lines through inductive electromagnetic interference. This was another reason to choose DC since it has the added benefit of having a $\sim$0 Hz frequency, or non-oscillating stability.

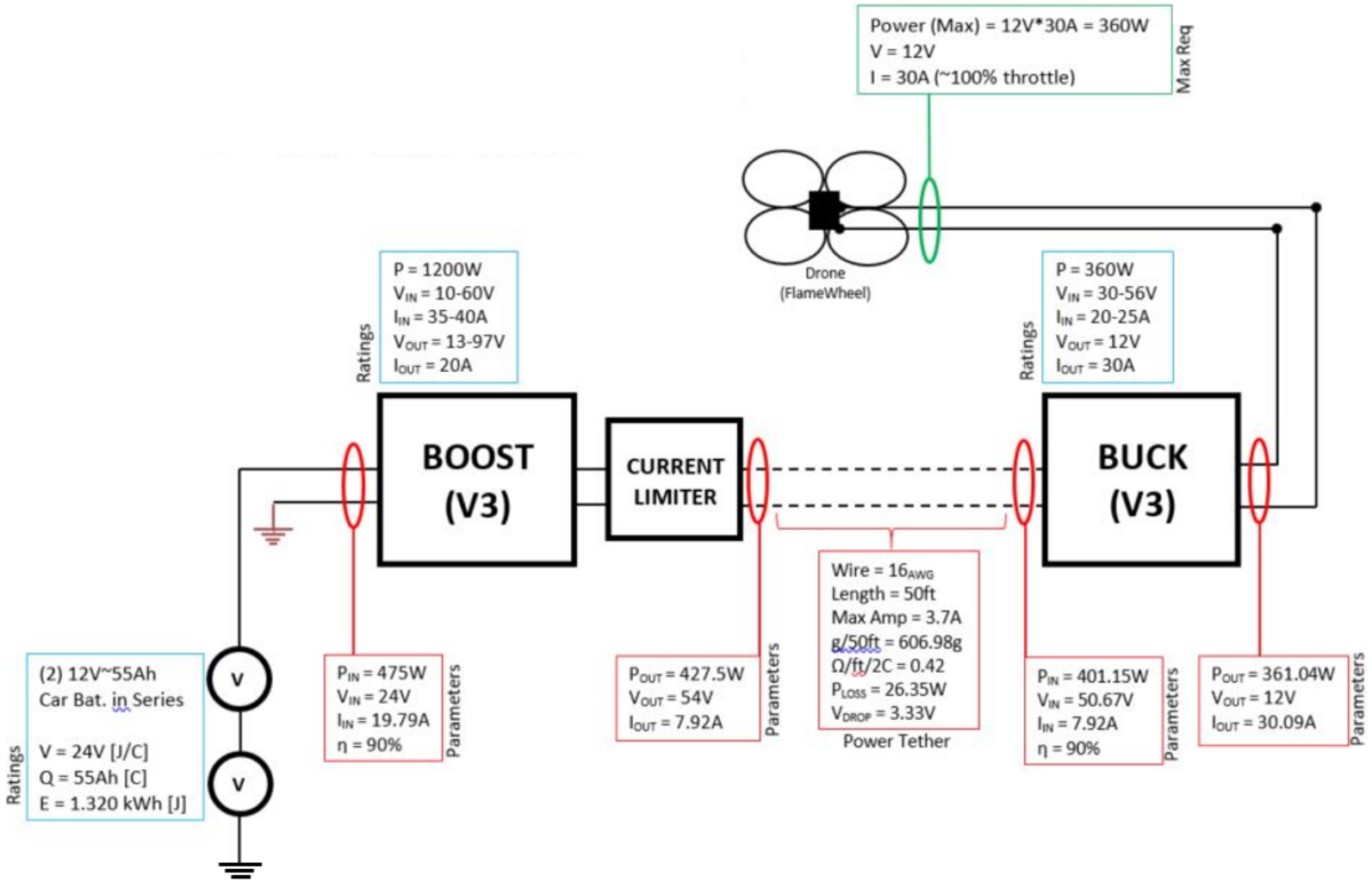


Figure 3.1: The TAUS-Alpha and Beta system level block diagram with an example of calculations for derived system parameters.

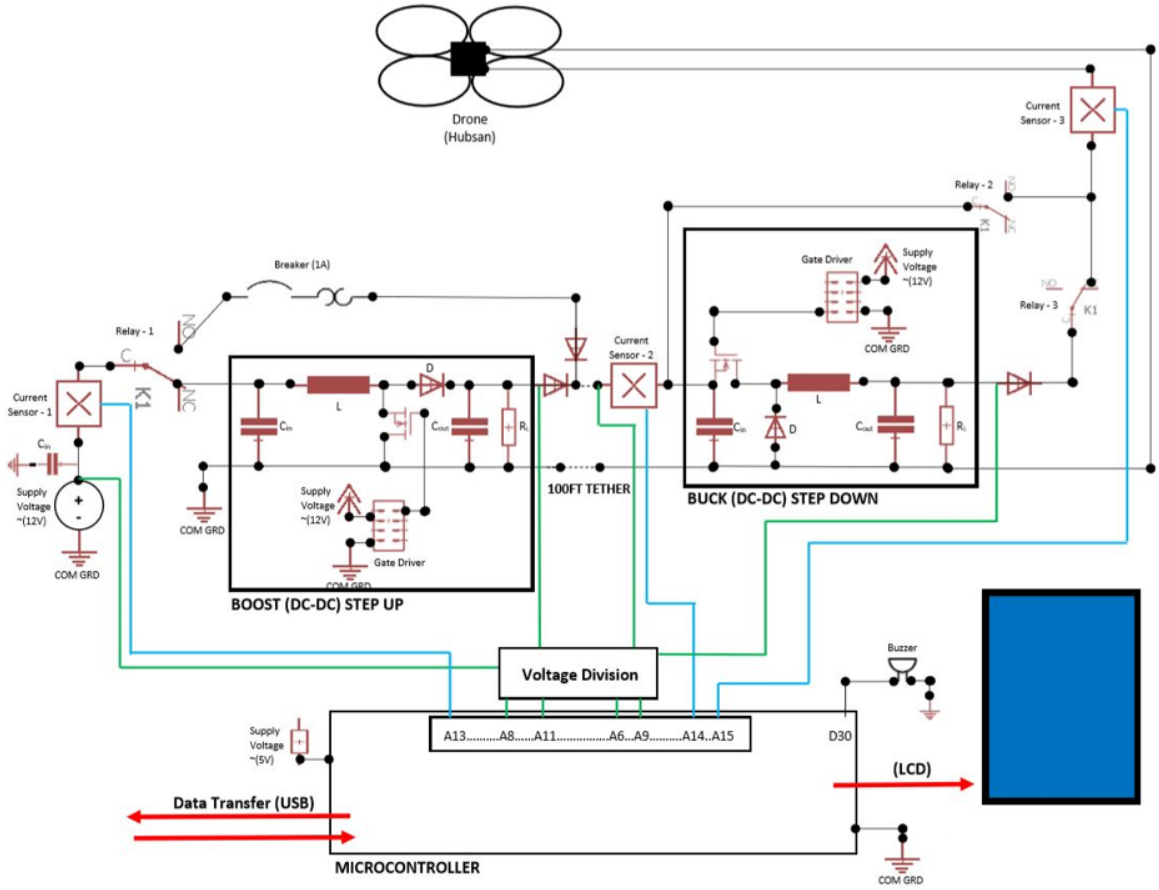


Figure 3.2: The TAUS-Alpha table-top prototype pseudo-circuit high-level diagram.

With these requirements in place, we then derived the other dependent system requirements that emerge based on the laws of physics expressed in electrical engineering theory (see Figure 3.1), such as system supply capacity, step-up/step-down power-electronics, transmission voltage, etc. Before developing a scaled version, we decided to test our theoretical values and approach by first developing a table-top version that can be analyzed.

3.2 Prototype - Alpha

With the electrical engineering theory established, we developed TAUS-Alpha, which is a small-scale table-top testbed for power-over-tether UAS. This version of the system was the first step toward answering Research Question 1 by allowing us to physically prototype the power-electronics hardware and system sensor software for a less demanding but representative version. It should be noted that this version did not have temperature sensors integrated. Using the system-level view of Figure 3.1 we developed the pseudo-circuit level view seen in Figure 3.2. TAUS-Alpha is composed of an ATmega2560 16MHz microcontroller, a low-end COTS boost step-up circuit, 100ft of 2-conductor 24 AWG copper multi-stranded cable, buck step-down circuit, a Holy Stone micro quad-rotor UAS, and numerous voltage and current sensors (see Figure 3.3). We used TAUS-Alpha to validate theoretical values and performance expectations before scaling up to TAUS-Beta.

The TAUS-Alpha takes 12V nominal input and steps up the DC voltage to 35V for transmission across a 2-wire 100ft 24 AWG tether to be stepped down to 5V DC to deliver adequate power to the micro-UAS load. Without stepping up the transmission voltage the resistive losses in the conductors would degrade the transmission voltage and generate too much current and therefore heat for the tether. The 32-gram micro-UAS is typically powered by an on-board 11-gram single cell LiPo cell (3.7V 380mAh nominal). We designed and 3D printed a simple telescopic scope for the UAS to be fixed and controlled testing. The microcontroller was responsible for interfacing with sensors, controlling relays, and pushing information to a Liquid Crystal Display (LCD). In-line hall-effect current sensors were integrated at system input, post-boost circuit, and post buck circuit. Voltage tap-ins were placed at the input, post-boost circuit, pre-buck circuit, and post-buck

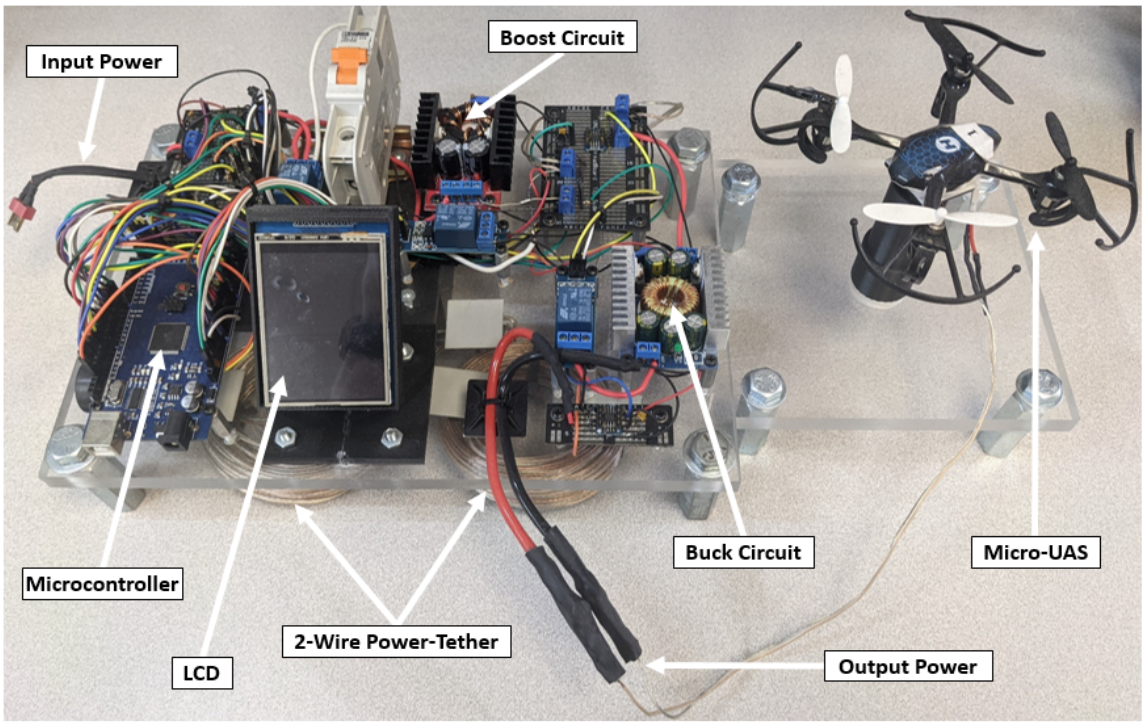


Figure 3.3: The TAUS-Alpha table-top prototype used to demonstrate the theory, circuits, and control systems necessary to scale to Beta. Major system components are labeled.

circuit which fed into individual voltage division circuits to transform voltage levels for analog-to-digital (ADC) pin tolerance on the microcontroller for interpretation. There are two power pathways through the system which are dictated by electro-mechanical relays controlled by the microcontroller. One path bypasses the boost and buck circuit connecting the input power directly to the UAS load via the tether without any power transformations. This path represents the classic problem with power transmission at low voltage and its inadequacy in our situation. There is an in-line 1 amp circuit breaker used to protect the 2-wire power tether from harmful current drawn across the transmission due to the inadequate voltage level. The other power-pathway is the intended path which utilizes the boost and buck circuits.

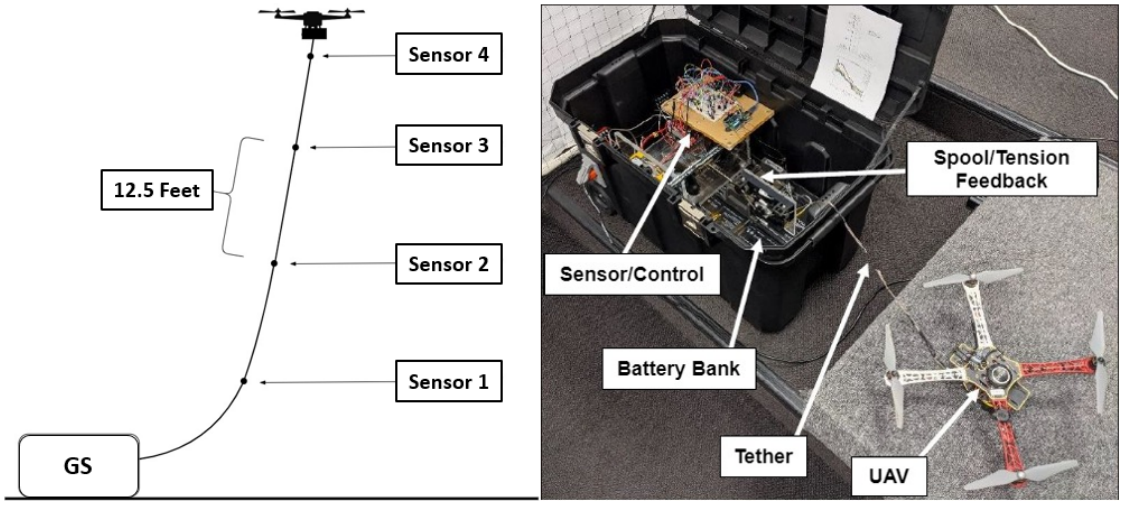


Figure 3.4: **(*Left*)** The TAUS-Beta high-level concept and direction. **(*Right*)** The MVP prototype developed in-house at UNL in the NIMBUS laboratory.

3.3 Prototype - Beta

From conducting our search for related work through the published literature and industry products, we discovered power-over-tether systems were already developed and no longer novel. We identified sensor array integration along the power-tether coupled with UAS trajectory as the novel research direction for our system. Leveraging the validation and information gained from TAUS-Alpha we scaled up to TAUS-Beta. This version was developed as a proof-of-concept minimum viable product (MVP) to helpfully answer Research Question 1. The high-level concept and physical MVP for the TAUS-Beta prototype can be seen in Figure 3.4.

3.3.1 System Composition

In this section, we will discuss the system's electrical, mechanical, and computational components. The TAUS is composed of three major subsystems: ground, tether, and aerial. The subsystems are physically linked via the tether subsystem.

The composition of each subsystem is presented in Table 3.1.

The TAUS is composed of three sub-systems: GS, Sensing-Power-Tether, and UAS. The GS contains a 24 volt 66 amp-hour capacity battery bank, DC-DC Boost power electronics that raise supply voltage to 56 volts, spooling feedback mechanism, system control circuitry, wireless communication, and sensor array bus interpretation. The sensing-power-tether is a jacketed pair that has DS18B20 sensors firmly integrated every 3.5 m. The UAS is a DJI F450. The onboard flight controller and associated peripheral devices are Pixhawk running standard ArduCopter firmware.

The ground subsystem is composed of a battery bank, power/sensor/control circuitry, and a mechanical spooling and tension feedback mechanism all packaged in a containment structure. The battery bank is 6 Mighty Max deep cycle 12 Volt (V) DC 22 Amp-hour (Ah) batteries compartmentalized and configured as 2-series 3-parallel (2S3P), as shown in Figure 3.5. This configuration allows for an overall nominal battery bank supply voltage of 24 V at 66 Ah of capacity. Since the primary electrical load of the battery bank is a UAS that is inherently weight limited and requires a DC supply, we decided to maintain DC throughout the entirety of the system. Advancements in power electronics allow us to easily and

Table 3.1: The TAUS-Beta prototype composition of major subsystem components.

Ground	Tether	Aerial
Containment Structure	4C 22 AWG (15m)	DJI F450 Flame Wheel
Battery Bank (2S3P)	DS18B20 Temp. Sensors	Pixhawk Flight Contr.
DC-DC Boost Circuit		Pixhawk Peripherals
Sensor/Control Circuit		Futaba RC Rec. (2.4 GHz)
Spooling Mechanism		DC-DC Buck Circuit
Tension Feedback Mech.		

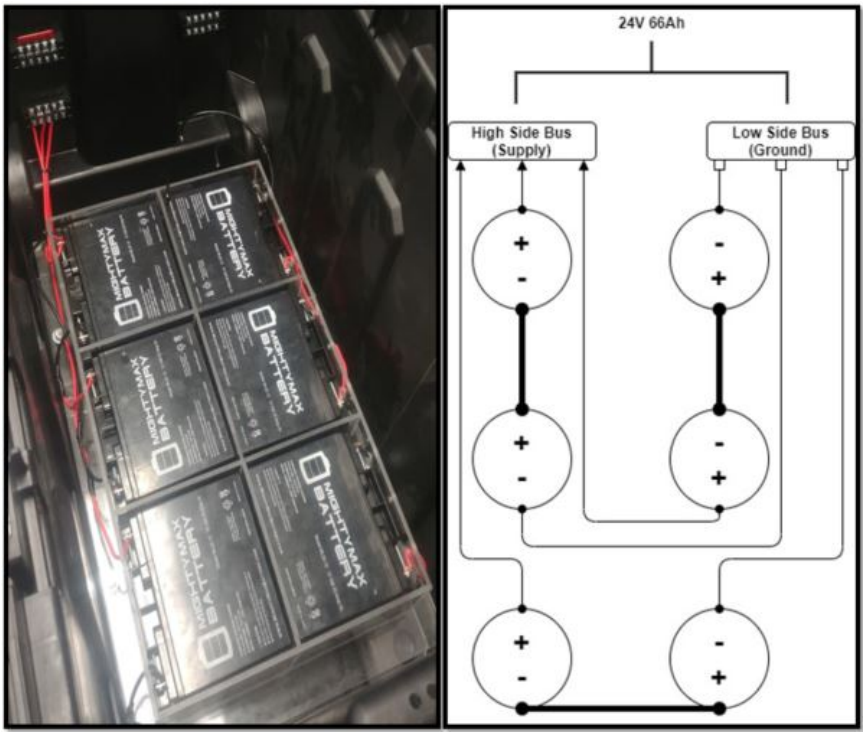


Figure 3.5: **(*Left*)** The TAUS-Beta physical sealed-lead-acid battery bank. **(*Right*)** The 2-series 3 parallel circuit diagram.

efficiently modulate DC voltage, providing us the opportunity to avoid heavy power transformers and rectifier circuits. Therefore, an adequate DC-DC step-up (Boost) circuit was integrated that could increase the voltage and source enough DC to the load across the tether. These electrical parameters were derived by first measuring the maximum power demands of the load and working power calculations backward to the source, which can be seen in Figure 3.1. At maximum throttle, the UAV demands a steady 12V and 30A or 360 Watts (W) of power. Therefore, it was determined that a Boost circuit that could raise the voltage to $\sim$55V and source a current of 8A was adequate for 50ft of the tether.

A custom spool was coupled with a slip ring and mountable bearing to allow for seamless electrical contact while rotating. The spool was designed in OpenSCAD and 3D printed on a Markforged-Mark Two printer with Onyx filament. The

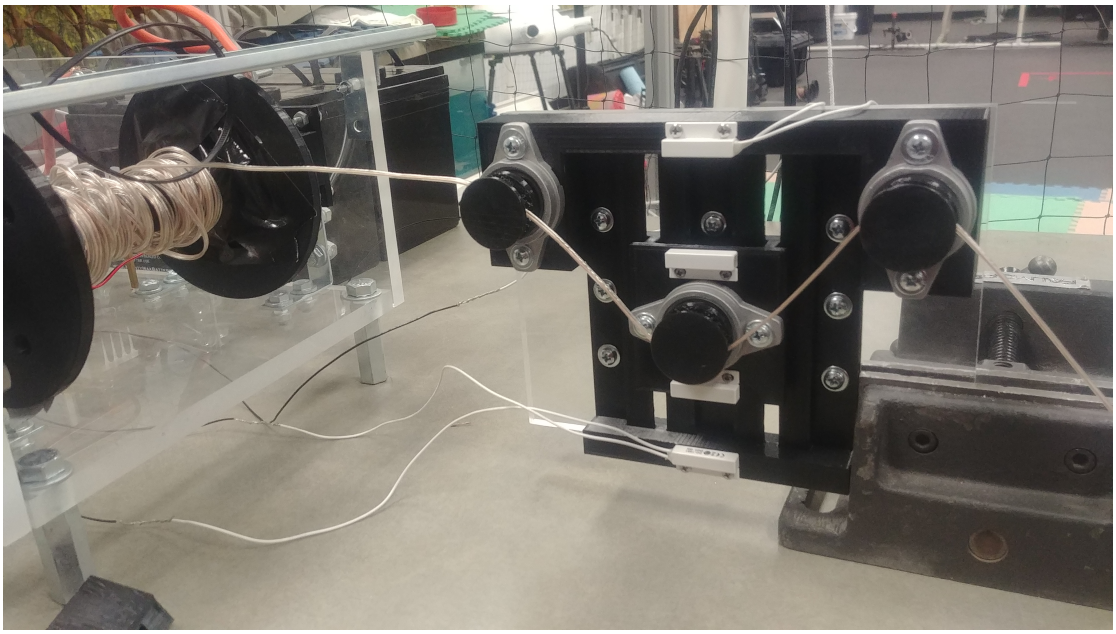


Figure 3.6: The TAUS-Beta tension feedback electro-mechanical system. The primary power-tether spool can be seen feeding tether to the system through a system of bearing rollers.

bidirectional nature of spooling was achieved with a high torque DC motor controlled by a high-power H-bridge circuit. The decision to either release or retract the tether from the spool is determined by a custom-designed tension feedback mechanism shown in Figure 3.6. The center section of the mechanism can actuate vertically and, in doing so, magnetically couple to one of two reed switches mounted on the device. The spool controller can exist in only 1 of 3 clearly defined states at any moment in time; therefore, control of the spool can be described as a rudimentary finite-state automaton. The 3 states from the tension feedback system are designated high-tension, low-tension, or explicitly neither state, a steady state. To interpret sensor data and control the various aspects of the ground subsystem, we integrated a microprocessor into the system. Specifically, we used a standard AVR architecture 8-bit 16-Mhz ATmega328P. The processor was programmed in C via the Arduino integrated development environment (IDE).

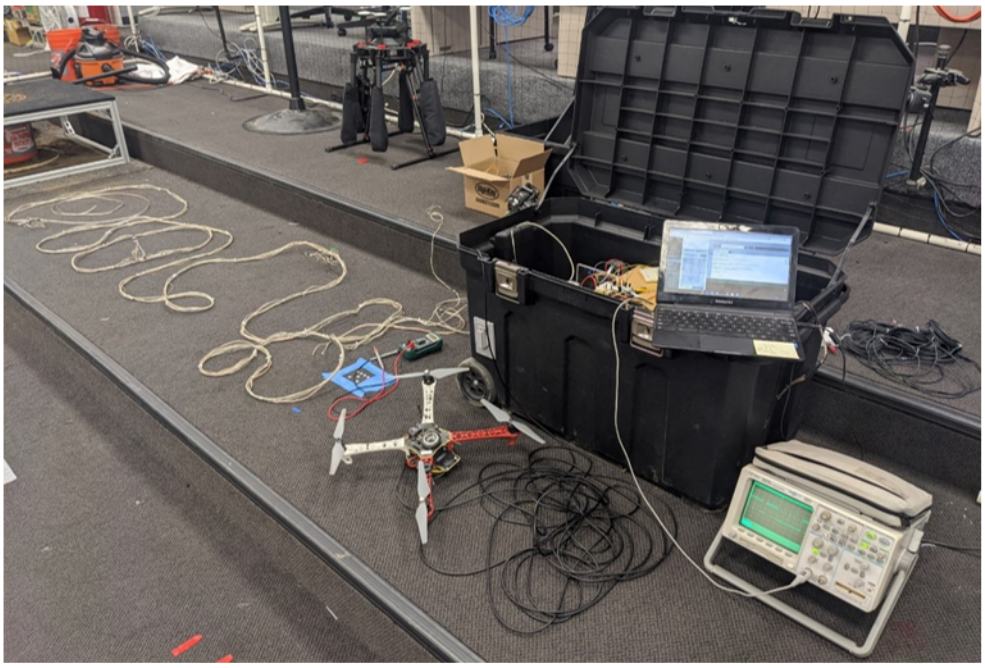


Figure 3.7: The TAUS-Beta initial power and sensor array integrated tether developed for laboratory testing.

The tether subsystem is composed of the physical power tether and the temperature sensor array along the tether. We decided to integrate the DS18B20 temperature sensor. This integrated circuit (IC) has a re-programmable resolution of up to 12 bits and can communicate via 1-Wire protocol at a nominal sampling rate of ∼0.750 Hertz (Hz). This is a unique form of open drain communication similar to inter-integrated circuit (I^2C), but instead of a designated clock and data line, these sensors rely on a single communication line. For proof-of-concept, we initially constructed the tether out of a jacketed 3-conductor cable for the sensor array and a jacketed 2-conductor cable for power transmission (see Figure 3.7) zip-tied together. Conductor shorts were prevented by a silicon coating to insulate the exposed sensor lines. To determine viability, an oscilloscope probe was connected to the data-line (see Figure 3.8). Since the data-line is considerably long (∼ 15m) the capacitive effect to swing from voltage rail-to-rail becomes more

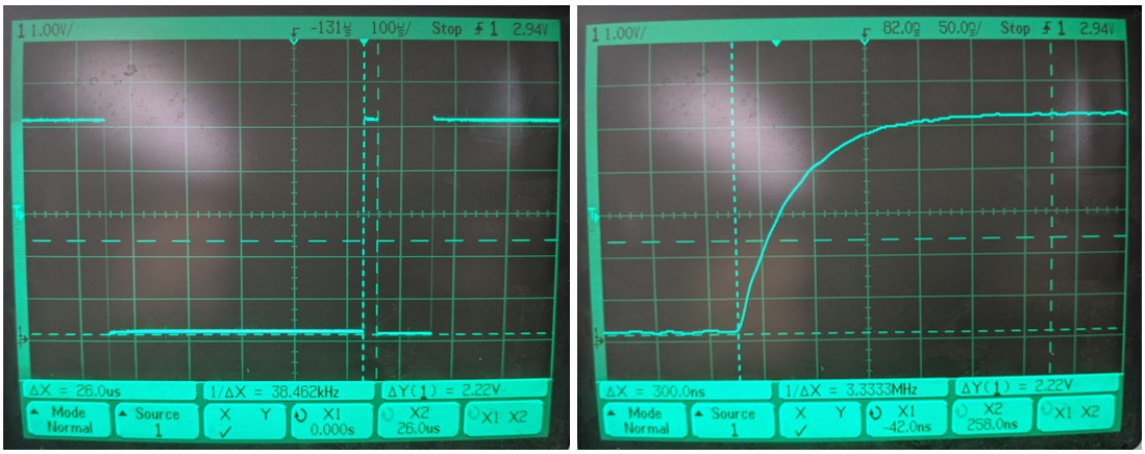


Figure 3.8: An oscilloscope showing a ds18b20 temperature sensor **(*Left*)** starting data bits and **(*Right*)** the capacitive effect on the voltage switching from rail-to-rail caused by a pull-up resistor open drain communication on a significantly long data cable.

apparent. To tackle this, we reduced the pull-up resistor's value from 4.7 Ω to 3.3 Ω. This increases the current through the resistor/sensors but not passed their ratings. When integrated with the microprocessor the data from the sensor array was accurately interpreted.

Since the amount of released tether from the spool contributes to the overall weight the UAV experiences, it benefits the overall system if the tether is maximum gauge and, therefore, minimum weight. The gauge and composition of the tether have implications for the power calculations previously discussed. In general, power transmission is proportional to an inverse relationship between the tether wire gauge and the transmitted voltage across it. If you want to decrease the tether, you need to increase the voltage. The reason is that the wire is rated for a particular continuous current, which if exceeded for a time, can result in the wire becoming thermally compromised and resembling the behavior of a fuse, creating an open circuit. With this in mind, we initially engineered the prototype to work with a 22-gauge tether. As discussed, limiting the weight of the tether is ideal; therefore,

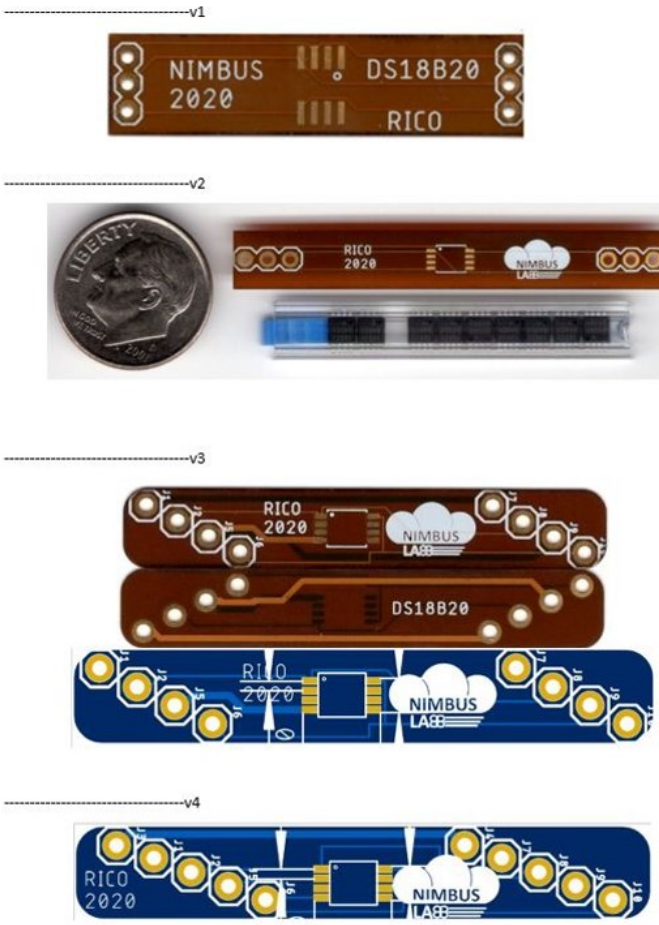


Figure 3.9: In order to spool the tether with sensors integrated in-line we initially thought it best to pursue flexible PCBs for the sensors. Versions 1-3 are flexible and version 3-4 is standard type. The sensor IC can be seen alongside version 2 with a dime for scale.

adding temperature sensors along the tether will increase the overall weight, particularly because of an additional communication line. If the sensors used require an independent communication line, implementation becomes impractical.

To make the tether more robust for spooling and field deployment we wanted to develop a means for the sensor array to be firmly integrated in-line. We designed a simple junction printed circuit board (PCB) in Eagle™to solder the sensor IC package to (see Figure 3.9). The PCBs were manufactured by Osh Park™[58] and

Figure 3.10: Version 2 of the flexible PCB with the sensor IC soldered in place.

Seeed Studio™[59]. Initially, we thought a flexible PCB would be ideal since the tether would be spooled but the lack of rigidity was not ideal, thus we reverted to typical form (see Figure 3.10).

We integrated sensors along the 4C tether in tapped-off parallel fashion on PCBs (see Figure 3.11). This placement scheme was determined based on high-sensor sampling resolution and the minimum possible natural atmospheric temperature gradient. A high-quality iMet temperature sensor was present on the ground alongside another DS18B20 sensor (Sensor 7). These ground sensors allow for a static undisturbed temperature reference to match the physically dynamic array's data against. When dealing with power transmission, power losses are inevitable due to quantum collisions as electrons flow along the conductors. These losses can be expressed as $P_{loss} = I^2R$, where 'I' is current (charge/time) in Amperes and R is the resistance in Ohms across the conductor. These collisions release energy in the form of heat, which increases the temperature of the conductor. To determine the effect this generated heat has on the sensor's ability to accurately sense the atmosphere, two sensors were placed at each position. One sensor is physically touching the tether influenced by conduction, and the other sensor is 4 inches off structure influenced by convection. Results derived from this sensor configuration

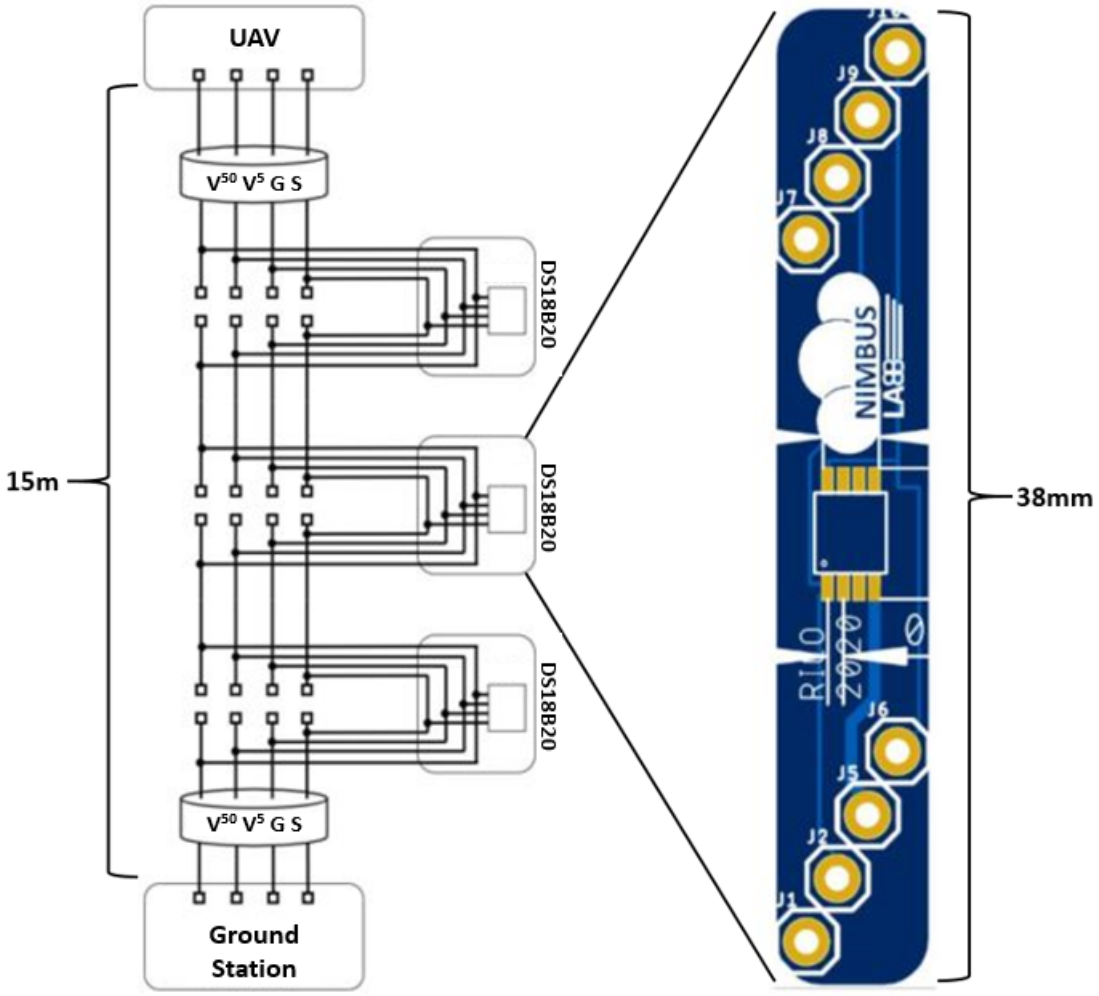


Figure 3.11: The printed circuit board developed for the atmospheric temperature integrated circuit sensor. The incorporation of the sensors along the power-tether along with scale are presented.

will dictate future sensor placement.

The aerial subsystem is composed of power circuitry and a COTS quadrotor UAV with a flight controller and associated peripherals. Specifically, the UAV is a DJI Flame Wheel F450 [60], and the coupled flight controller and remote control (RC) is a Pixhawk PX4 [61], and Futaba T6J, respectively. As discussed, the UAV is expecting a steady 12V DC supply, but at various low electrical load conditions, low throttle, the UAV will experience nearly all of the 55V stepped-up voltage

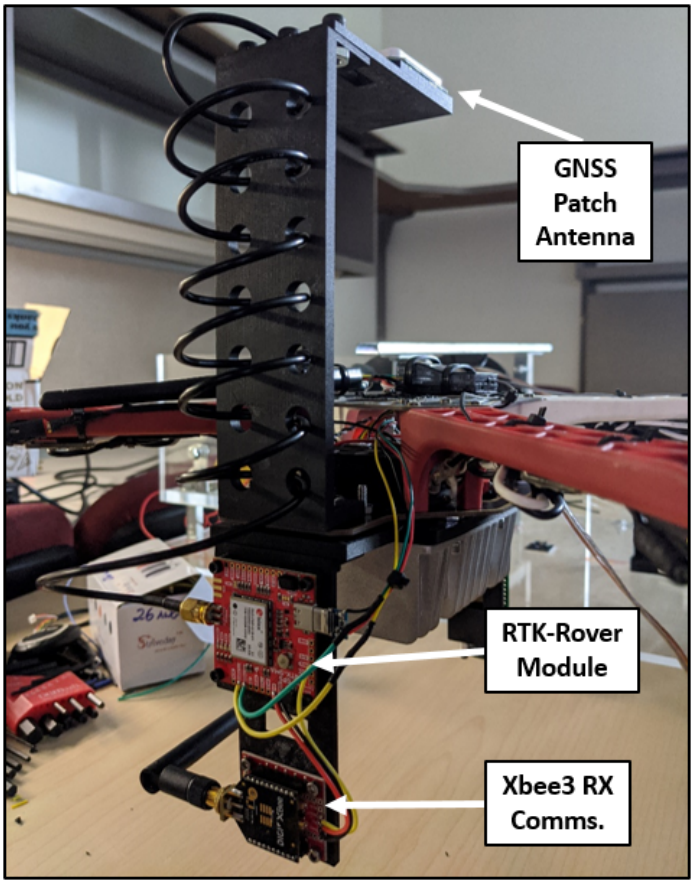


Figure 3.12: The RTK-GNSS Rover hardware integrated on-board the UAS.

from the Boost circuit. The UAV and flight controller onboard power electronics are not rated for such a high sustained supply voltage. Therefore, we integrated a DC-DC step-down (buck) converter circuit onboard the UAV that is rated for such input conditions but can still source up to 30A for max throttle. Upon complete physical construction of the TAUS prototype, we conducted outdoor experiments to assess the effectiveness of our system in the field.

To conduct accurate and precise trajectories with a physical system in unstructured and dynamic environments, we integrated and configured real-time kinematic (RTK) global navigation satellite system (GNSS) hardware onboard the UAS (see Figure 3.12). Typical satellite and barometer-based positioning have

significant variance on the order of meters, whereas RTK-GNSS differential positioning can be on the order of centimeters. Specifically, a set of u-blox ZED-F9P modules were configured for base-rover interaction. Module-to-module communication for the base to rover differential corrections was accomplished with a set of paired Digi Xbee-3 radios operating at 2.4 GHz via the 802.15.4 protocol.

3.3.2 Sensor Array Calibration

To inform our upcoming temperature sensor modeling, we empirically derived each sensor's intrinsic τ from an analysis of the calibration data. We carried out a two-point calibration on our sensor array to account for both slope and offset errors against an iMet-XQ2 bead thermistor. The iMet-XQ2 and array were provided exactly 1-hour to reach their respective steady-state equilibrium at a room temperature of 22.06 °C and at -15.30 °C in a freezer under aspiration (see Figure 3.13).

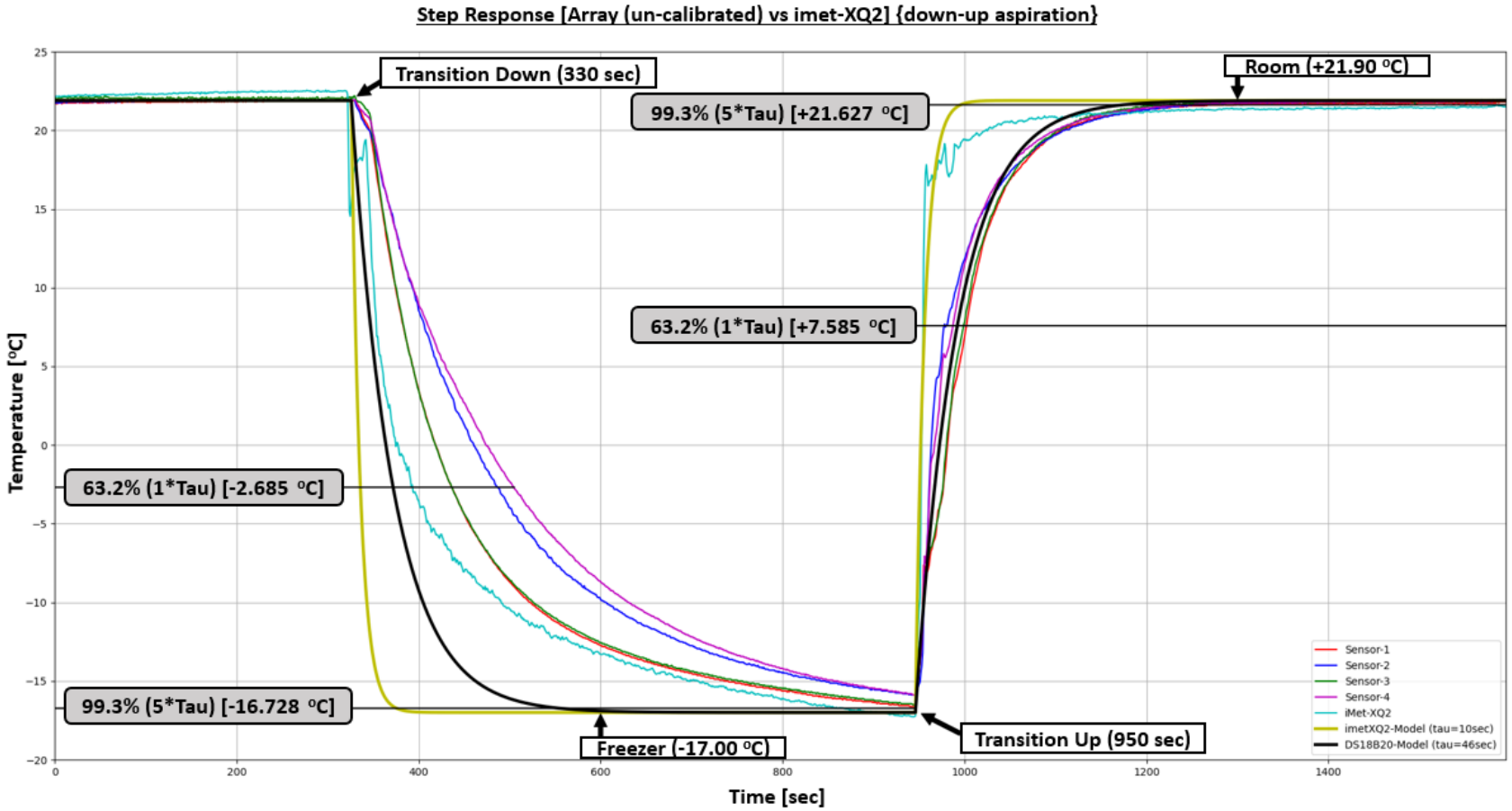


Figure 3.13: The temperature sensor step-response data that spanned 1600 seconds. Temperature values spanned from +22.06 °C to −15.30 °C. The sensor was under aspiration throughout the while exposed to room temperature and the freezer.

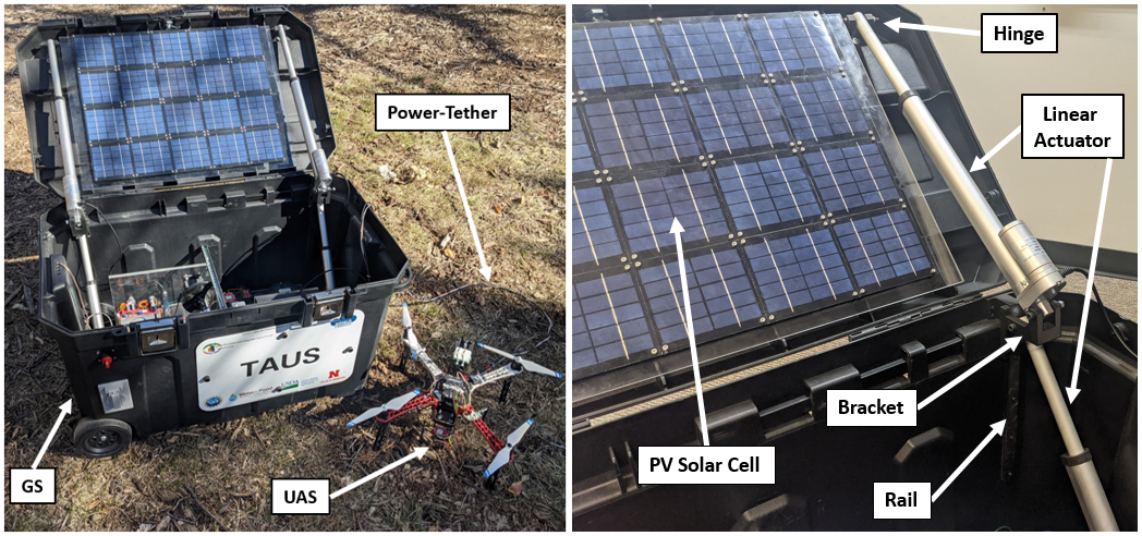


Figure 3.14: ***(Left)*** The TAUS-Beta shown as a retrofit system. The physical system's subsystems are labeled. ***(Right)*** An up close look at the retrofit linear actuators for autonomous lid operation along with a solar array for battery bank recharge.

3.3.3 Retrofit Upgrades

As the research progressed we identified various additions to the system that would enhance its field operation. Specifically, a step towards an ability to autonomously deploy, and recharge its battery bank in isolation from the electrical grid. To address this, we incorporated mechatronic linear actuators in a two-step erector configuration coupled with various hinge, bracket, and rail components all of which were designed and 3D printed in-house. We also integrated a 40W 2S1P configured photovoltaic solar array which was recessed in the lid of the ground station 3.14. The specifications of an individual solar cell can be seen in Table 3.2. We coupled the photovoltaic array with a solar charger to manage the battery bank recharge schedule and profile. It runs a Maximum Power Point Tracker (MPPT) algorithm, which is a type of Hill-Climbing sub-optimal controller [62] and an industry standard. These retrofit upgrades can work together by actuating the lid to both autonomously deploy the system in-situ and to find the optimal angle to

Table 3.2: Photovoltaic cell parameters and values at standard testing conditions (STC) from High Quality Solar Technology (HQST) manufacturer.

Parameter	Value (at STC)
Max Power at STC (P_{max})	10 W
Open-Circuit Voltage (V_{oc})	21.5 V
Optimum Operating Voltage (V_{mp})	17.4 V
Optimum Operating Current (I_{mp})	0.578 A
Short-Circuit Current (I_{sc})	0.620 A
Temp. Coefficient of P_{max}	-0.44 %/^{o}C
Temp. Coefficient of V_{oc}	-0.30 %/^{o}C
Temp. Coefficient of I_{sc}	-0.04 %/^{o}C
Max System Voltage	600 VDC
Max Series Fuse Rating	3 A
Fire Rating	Class C
Weight	1.0kg/2.2lbs
Dimensions	340mm x 240mm x 17mm
Standard Test Conditions (STC)	Irradiance 1000 W/m^2, T = 25 ^oC, AM=1.5

solar exposure which will help maximize each day's solar hours and battery bank recharge.

Chapter 4

Field Experiments & System Validation

This chapter aims to answer Research Questions 2 and 3 stated in Chapter 1: ”**Will the power transmission efficiency bias the atmospheric temperature sensors?**” and ”**Can we validate the systems base-level operational performance in the field?**” based on our *hypotheses*: ”If the power transmission efficiency is below a certain unknown percent the power-tether conductors will experience significant electron-based quantum collisions and that will result in excessive heating which will radiate through the proximal atmosphere biasing the sensors.” and ”If we take the system to the field to conduct a quasi-static sampling experiment in proximity to a well maintained and quality ground-truth field tower installation we will be able to run an intercomparison analysis to determine sampling error between the two at altitude.”

This chapter is organized as follows: experimental setup, Beta field experiments, system performance, and analysis of the subsequent field results are described in Section 4.1, 4.2, and 4.3, respectively.

4.1 Experimental Setup

We looked to first test the system's ability to fly for significantly longer times than typical COTS systems and to simultaneously sense the atmospheric temperature from the array of sensors integrated along the tether in-line and off-line. This sensor configuration allowed us to determine the impact of power transmission efficiency on the sensors' ability to sense atmospheric temperature accurately without bias. Taking into account the capacity of the battery bank, average demand of the load, and a COTS system average flight time of 30min, we decided on a field test of 6 continuous hours or a 12-fold increase at an AGL of 15 meters would be a satisfactory demonstration and benchmark. When carrying out the test, we used an autonomous flight mode on the flight controller that attempted to hold a particular GPS position. We also used an onboard barometer to attempt to hold a particular altitude. The experiments were conducted at the University of Nebraska - Lincoln's Havelock Research Facility on North 84th Street and Havelock Avenue in Lincoln, Nebraska, USA.

Regarding a field test to determine the accuracy of the sensor array, we decided to run a quasi-static hour-long 15 meter AGL test against a 10m AGL well-maintained MESONET tower. The experimental data were acquired from the TAUS-Beta and the tower installation for intercomparison error analysis to determine system validation.

4.2 Sensor Array Bias

The long-term 6-hour continuous flight field test ran to completion and was therefore determined a success. Since the test was carried out autonomously, it should be noted that a human had to periodically intervene to reestablish

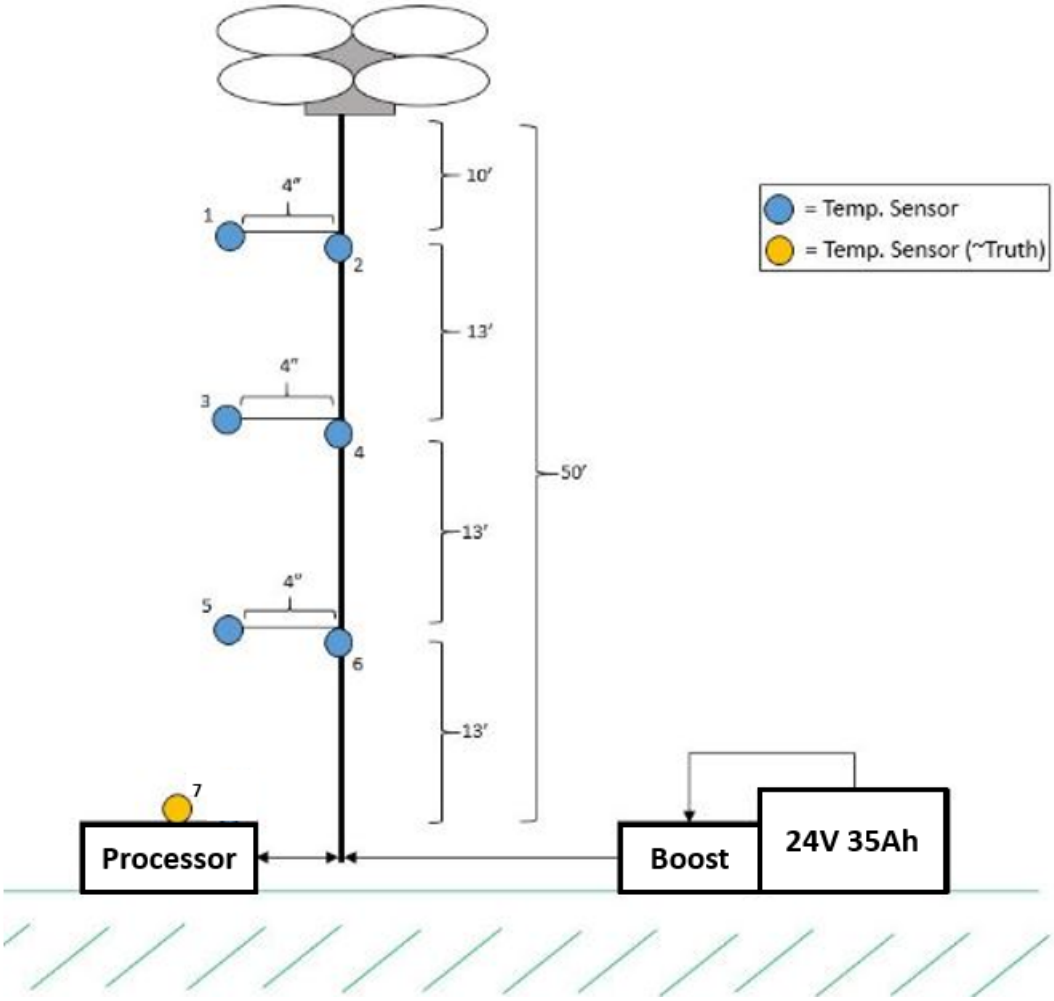


Figure 4.1: A high-level diagram of the TAUS-Beta temperature sensor array positions for array's both on and off the physical power-tether. This configuration was used to determine the impact power transmission losses to heat have on the sensors.

an appropriate AGL. This drift is to be expected due to changing atmospheric pressure throughout the test and the inherent limitations of a single, onboard reference barometer. The second test previously described also ran to completion and successfully acquired atmospheric temperature data from the sensor array along the tether shown in Figure 4.1. It was expected that when the temperature data acquired from the sensor array was plotted, we would see a decreasing

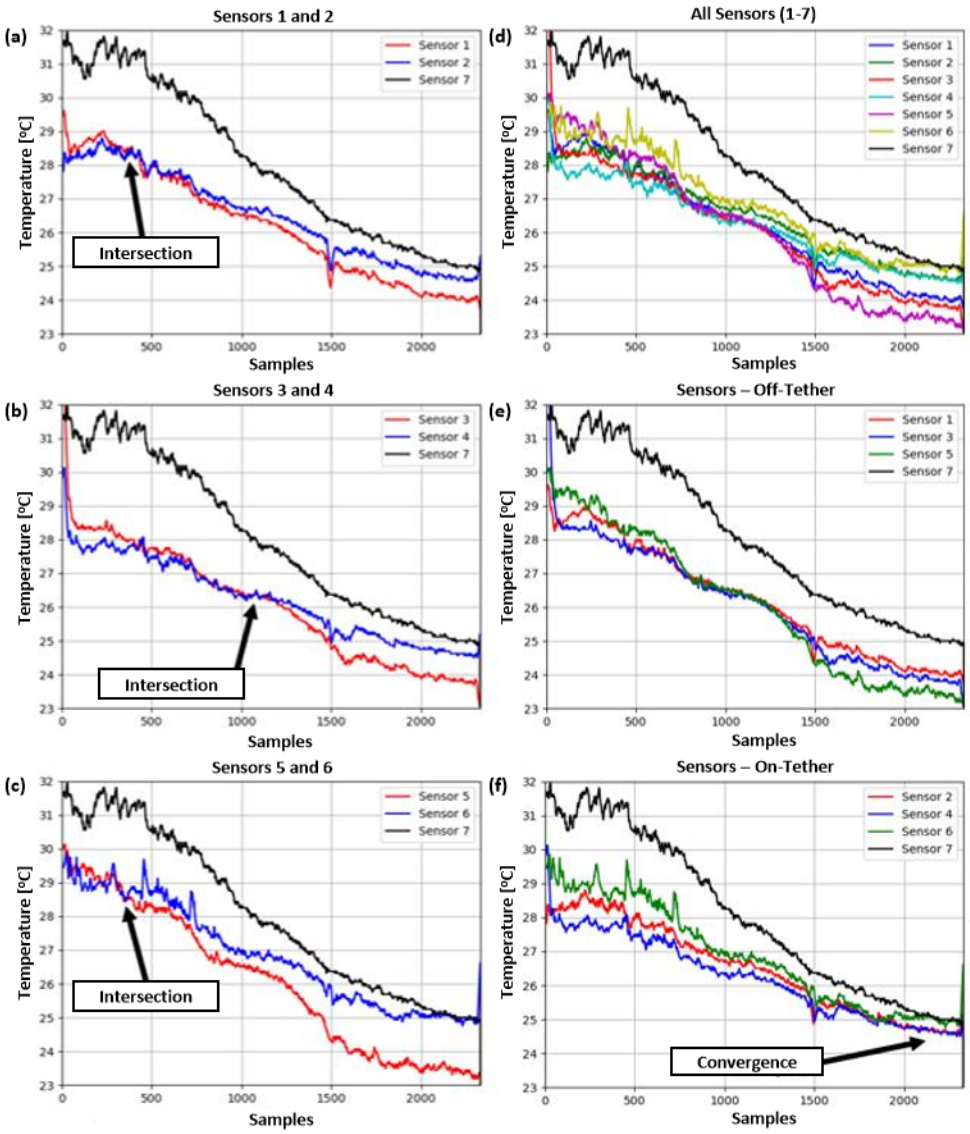


Figure 4.2: The temperature data time-series from the on-off power-tether field experiments at the UNL-Havelock Research Facility at sunset. **(a,b,c)** shows each set of sensor pair (on-off) related to Figure 4.1. **(d)** shows all of the sensors in one graph and how they relate. **(e,f)** shows all sensors on and all sensors off power-tether, respectively.

trend due to the setting sun, and that is exactly what we see in Figure 4.2. We experimented for approximately 1 hour at sunset. The array of sensors were able to collect 2,389 raw samples where 60 (2.5%) were erroneous. Thus, the total net number of samples acquired from the array was the difference between the raw

and erroneous samples, which is 2,329. Considering the raw number of samples, the array sampled at a rate of 0.664 samples per second (0.664 Hz), or 39.82 samples per minute. As previously stated, the anticipated sampling rate for each sensor and, therefore, the array is ∼0.750 Hz. This is a difference of 0.086 Hz meaning the array sampled 11.47% slower than expected. This divergence may be explained by the unique sensor configuration and nature of open-drain digital communication. The communication line is pulled up to 5V through an external 3.9 Ohm resistor. When the data line is dynamically returning to a high steady-state, it must first load the parasitic capacitance of the line through the resistor. The time it takes to do this is expressed as (tau = RC,) where R is the pull-up resistor in Ohms and C is the load capacitance of the communication line in Farads. Since our configuration places sensors along a relatively long communication line, the parasitic capacitance effect and the multiplicative term becomes more influential. This delay may have propagated throughout the field experiment, resulting in the reduced sampling rate we observe. The immediate differences observed between Sensor 1-6 and Sensor 7 from the data can be largely explained by sensor physical distance to the ground. The surface of Earth absorbs the sun's shortwave ultraviolet (UV) electromagnetic radiation heating and radiating away via long-wave IR radiation after a latency period. This clear differentiation we observe is presumably due to this effect. It should be noted that at ∼1500 samples into the experiment, a human had to intervene and correct the TAUS's AGL. The sharp disturbance that can be observed in the data is a result of this adjustment taking place.

When looking closer at Figure 4.2 (a), (b), and (c) we can clearly see points of intersection where two sensors (excluding Sensor 7) exchange profiles. In all three graphs, the sensor that was on-tether initially sensed lower atmospheric temperatures relative to its off-tether counterpart. This is more clearly represented

Table 4.1: Statistical analysis on atmospheric temperature data for 4.2 (e) and (f).

Statistic	Off-Tether [^oC]	On-Tether [^oC]
Maximum	29.56	29.75
Minimum	23.25	24.56
Range	6.31	5.19
Mean	26.08	26.52
Median	26.25	26.44
Mode	24.00	25.06

in Figure 4.2 (e) and (f) and Table 4.1 where all three off-tether and on-tether sensors are presented together, respectively. Early on through the experiment, we see the behavior flips and the on-tether sensors report higher atmospheric temperatures through the duration of the experiment. This behavior may be explained by the influence of the power-tether. Specifically, before the intersection happens, the tether may behave as a heat-sink through conduction to the sensor, and after the intersection, as the experiment progresses through time, the tether behaves more like a heat source. The power losses across the tether would heat the conductors in the tether over time until potentially reaching an equilibrium. Even at equilibrium, the tether would still steadily radiate heat so we should expect the sensors reported temperature data to level off. If we look at 2000 samples onward in Figure 4.2 (a), (b), (c) and (f), we can see the on-tether sensors head toward a consensus like horizontal asymptote around 24.75 degrees C. If we extrapolate the data further in time, we would expect to see a clear intersection point between all on-tether sensors and Sensor 7, indicating the influence of the power-tether, specifically the power losses have saturated the sensors' ability to accurately sense the atmospheric temperature. To prevent this negative effect, the efficiency of the power transmission across the tether throughout the experiment must be relatively

high.

The prototyped TAUS has been successfully field-tested for base functionality, and therefore, we have demonstrated the proof of concept. We uploaded to YouTube a set of videos that track TAUS developments and field experiments through time [63] and provide a high-level overview [64]. The system performance through these two field tests has been evaluated from an empirical perspective. After the tests were complete, we conducted a thermal inspection on the system hardware with an infrared (IR) thermometer. Moderately elevated component temperatures were measured, invoking little concern. The system was immediately re-deployable, showing system reliability. With the successful demonstration and evaluation of increased flight times, coupled with an ability to acquire data from the sensor array along the tether, the hypothesis presented has been tested and verified as a viable solution to our research question.

4.3 System Validation

The operational performance of the TAUS was tested against a national weather service (NWS) meteorological station tower (call sign: Eagle_3NW) located at the University of Nebraska-Lincoln's Rogers Memorial Farm [65]. (40° 51′ North, 96° 28′ West, and 370.332 m mean-sea-level (MSL) elevation). The tower is 10 m above ground level (AGL) with Vaisala HMP155A ambient temperature sensors located at 2 m AGL and 10 m AGL (see Figure 1.1). Both sensors are configured to sample at 0.2 Hz and log the average of 12 atmospheric temperature samples captured over 60 seconds. In between sampling and logging, the sensor is still exposed to the thermodynamics of the atmosphere. We considered both sensors a form of ground truth to match our system against. The experiment was conducted from

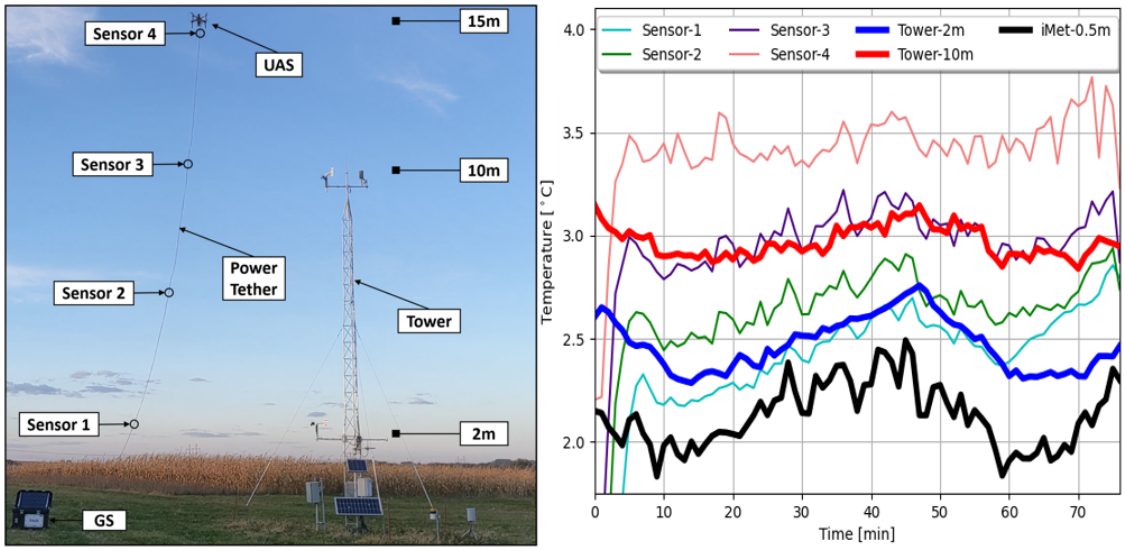


Figure 4.3: ***(Left)*** TAUS-Beta field experiment at Rogers Memorial Farm. ***(Right)*** Temperature sensor array vs tower sensor data at 10 m and 2 m at sunrise. The relationship between red-purple and blue-cyan is of special interest.

6:19am to 7:36am CST on Friday, October 16, 2020, and the average wind speed throughout the experiment was 1.77 m/s.

We acquired a total of 77 points per tower sensor (one per minute). Each of the four temperature sensors on the sensing-power-tether for the TAUS was configured to sample at 1 Hz acquiring ∼18,480 raw ambient temperature samples (one per second). We down-sampled and smoothed the system data to equate the number of points acquired by the tower sensors (see Figure 4.3). Now, with the same number of degrees-of-freedom (77 points) the temperature data for the TAUS and tower sensors were compared statistically (see Table 4.2). Since the system's sensors are exposed near the ground before the experiment they need a period to adjust to their new altitude-based ambient temperature. To account for this the analysis was conducted on a subset of the data that spanned 7min (+7) to 74min (-3) for a total of 67 degrees of freedom for analysis. We can infer that Sensor_1, Tower_1, and Sensor_3, Tower_2 with respect to each pair of data sets

Table 4.2: Statistical analysis of mean (μ), median ($\mu^{\frac{1}{2}}$), standard deviation (σ), variance (σ^2), and root-mean-square error (RMSE) for the data acquired by the TAUS array, iMet-XQ2, and tower sensors at 2m (T_1) and 10m (T_2). Minimal error between array and tower sensors are in **bold**.

Data Set	μ	$\mu^{\frac{1}{2}}$	σ	σ^2	RMSE-T_1	RMSE-T_2
$Sensor_1$	2.450	2.467	0.156	0.024	**0.143**	0.530
$Sensor_2$	2.667	2.657	0.114	0.013	0.232	0.309
$Sensor_3$	2.991	2.997	0.108	0.012	0.538	**0.091**
$Sensor_4$	3.450	3.431	0.096	0.009	0.999	0.500
$iMet_{XQ_2}$	2.123	2.120	0.163	0.026	0.356	0.849
$Tower_1$	2.463	2.447	0.134	0.018	00.00	0.505
$Tower_2$	2.963	2.938	0.075	0.006	0.505	00.00

have minimal error of 0.143 °C and 0.091 °C. Since both sets of sensors were at comparable altitudes, this outcome reinforces our expectations. Additionally, we can see a natural altitude based ambient temperature gradient when inspecting μ and $\mu_{\frac{1}{2}}$ for $Sensor_1$ up to $Sensor_4$, and $iMet_{XQ2}$ up to $Tower_2$. We can conclude that there is minimal sensing error at the corresponding altitude relative to the ground truth installation.

Chapter 5

Trajectory Simulations & Evaluation

This chapter aims to answer Research Question 4 stated in Chapter 1: "**Can we evaluate a set of system trajectories against well-defined metrics to determine operational performance?**" based on our *hypothesis*: "To evaluate the systems theoretical operational performance we will need to computer model and simulate the system in a physics emulated environment."

This chapter describes the design constraints and spatial bounding imposed by the system being physically tethered in Section 5.1, the model development process dictated by the constraints in Section 5.2, determination of the simulated trajectories in Section 5.3, the well defined factors to evaluate the simulations in Section 5.4, and the static and dynamic temperature field reconstruction is described in Section 5.5.

5.1 Model and Simulation Constraints

Unlike traditional UASs the TAUS is physically constrained by a sensing-power-tether. Thus, we need to quantify the systems' viable spatial bounds and traversable volume (see Figure 5.1b). The length of the tether L_{tether} the TAUS has is 15.375 m, and it has four atmospheric temperature sensors firmly integrated every 3.5 m

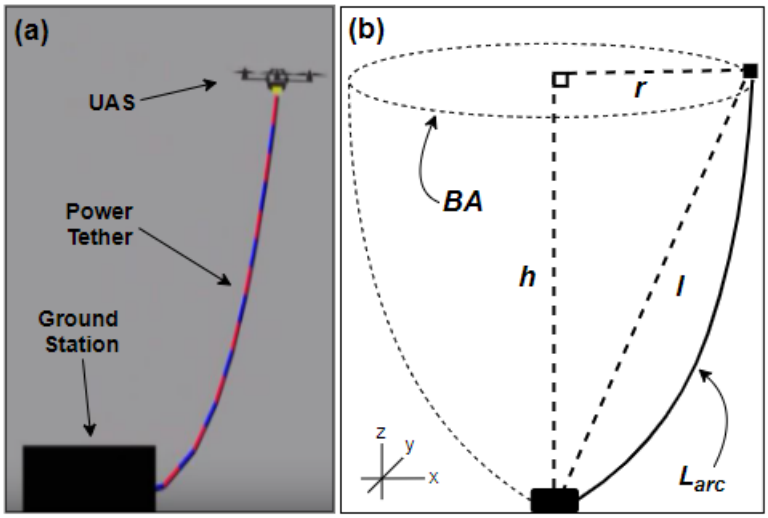


Figure 5.1: **(a)** The model developed for simulations. **(b)** Graphical representation of the traversable volume and bounded spatial extent.

starting at 4.85 m from the ground station (GS). The tether is not actively tensioned meaning it will express varying levels of both taut and slack behavior throughout the traversal. Therefore, the tether assumes the shape of a catenary which is formally defined as a hyperbolic cosine function, $f(s)$, suspended between the origin and a defined maximum distance. The length of this curve L_{arc} equals L_{tether} such that:

$$L_{arc} = \int_a^b \sqrt{1 + (f'(s))^2}\, ds \tag{5.1}$$

where, the lower bound is at the origin ($a = 0$) and the upper bound b is the maximum lateral spatial extent value ($b = r$). The lower bound, a, was set to origin of the Cartesian plane. As an approximation for the shape of the curve, the function $f(s)$ can be simply expressed as a monomial $s^{1<w\leq n}$, where w is a non-negative exponent value that represents the catenary behavior and n is an unknown upper bound. We set $w = 2$ producing a quadratic for parabolic behavior and therefore used s^2, where the first derivative of $f(s)$ is simply $2s$. Since the L_{arc} (which equals

L_{tether}) in Equation 5.1 is known and fixed, we can evaluate the integral to find the upper limit. Doing so, we obtain r = 3.814 m. Hence, the maximum lateral extent is $x = \pm 3.814$ m and $y = \pm 3.814$ m. Our parabolic approximation resolves the height bound, h, to 14.543 m. Thus, the TAUS was calculated to be bounded in three-dimensional space ($x = \pm 3.814$ m, $y = \pm 3.814$ m, $z = +14.543$ m). The slack in the tether at maximum spatial extent is the difference between the L_{arc} and the hypotenuse $l = \sqrt{r^2 + h^2}$, which is 0.357 m.

With the maximum spatial bound defined, our trajectories have a region of viable operation. Trajectories are inscribed entirely within the bounded area (BA), which is described simply by πr^2. To evaluate the trajectory's effectiveness at sensing a particular bounded region, we calculated the maximum traversable volume. The geometric shape that approximates the maximum spatial bound is a parabolic cone or paraboloid. The volume of an ideal paraboloid (V_{par}) can be expressed and evaluated with respect to h as:

$$V_{par} = \int_0^h \pi v \, dv \; = \frac{\pi v^2}{2} \Bigg|_0^h = \frac{1}{2}\pi h^2 \tag{5.2}$$

5.2 Model Development

To accurately represent the TAUS in a physics emulated environment to simulate system trajectories, we need to develop a model of TAUS-Beta. We modeled the physics of TAUS in flight with Simulink®. We based this on an existing high-fidelity 6 degrees-of-freedom quad-rotor UAS simulation system [66] and modified it to incorporate the characteristics of TAUS (see Figure 5.1a). The power tether was modeled as a series of cylindrical objects and universal joints. Each object contributed proportionally to the model's simulated weight for the tether, and each

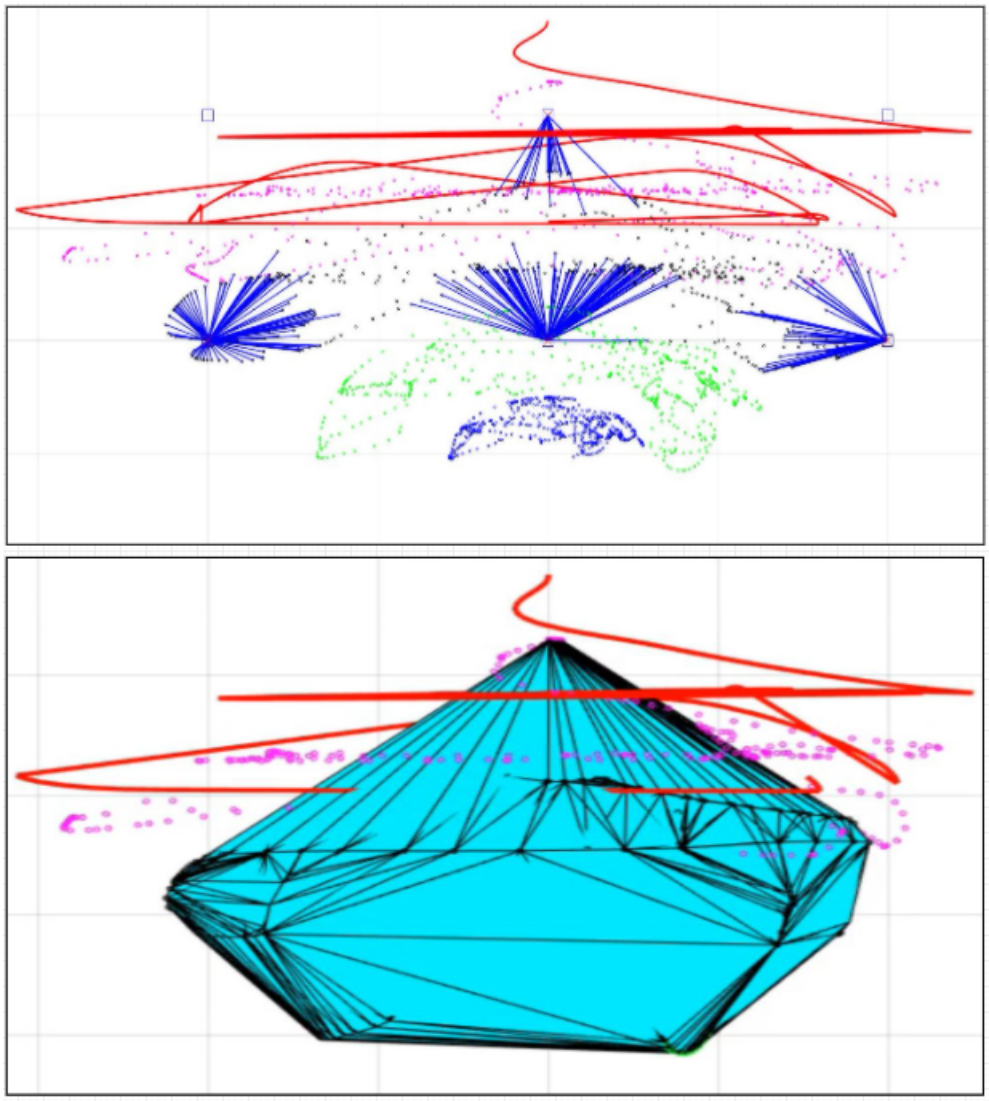


Figure 5.2: ***(Top)*** An example plot of the spatial information for the sensors (multi-colored dots) and UAS model (red) derived from the TAUS simulations. ***(Bottom)*** The volume was derived from the development of a convex hull from the sensor spatial data.

universal joint had spring stiffness and damping properties that were modified to emulate the physical power-tethers' elastic behavior. A total of 30 links at 0.02 m diameter and 0.5 m length were coupled with 15 universal joints to create the tether model. The quad-rotor UAS has previously been employed to model UAS suspended tethers acting under the surface of water bodies [67], and spinning

tethered pendulum for payload exchanges [68]. In addition, we mathematically characterized the DS18B20 temperature sensor as a linear time-invariant (LTI) first-order differential equation:

$$\begin{aligned} \tau\frac{d\psi}{dt} &= -\psi + \psi_c; \quad \psi(0) = \psi_0; \\ \implies \quad \psi(t) &= (\psi_c - \psi_0)\cdot(1 - e^{\frac{-t}{\tau}}) + \psi_0 \end{aligned} \tag{5.3}$$

where $\psi(t)$ is the sensor's temperature at time t, ψ_0 is the initial temperature, and ψ_c is the current ambient temperature the sensor is attempting to sample. We empirically derived the integrated circuits intrinsic time constant (τ) and step response time ($5{\cdot}\tau$), which is the time it takes the sensor to reach $1 - e^{-1}$ (63.2%) and 99.3% of the step, respectively. This was accomplished by placing the physical sensors in temperature-controlled environments with appropriate levels of aspiration for convective heat exchange (see Figure 3.13 as an example). The resulting time-series data from the sensor asymptotically reached a steady state with the control. Thus, the τ and $5{\cdot}\tau$ values were acquired. With this mathematical representation, the sensors were modeled and incorporated in the simulations to allow for real-time sampling and reconstruction of static and dynamic temperature fields across a bounded volume.

5.3 Trajectory Determination

Traditionally, full spatial coverage traversal is accomplished by invoking an exhaustive walk ***Lawn-mower*** trajectory [69], also classified as a scan back-and-forth (BF) creeping line. We, therefore, select Lawn-mower as one of the candidates and additionally propose three more patterns (see Figure 5.3). These are (2) ***Archimedean Spiral*** that provides a prolonged yet high spatiotemporal resolution sampling; (3) ***Pentagram (Star)*** that provides a quick yet low spatiotemporal resolution re-

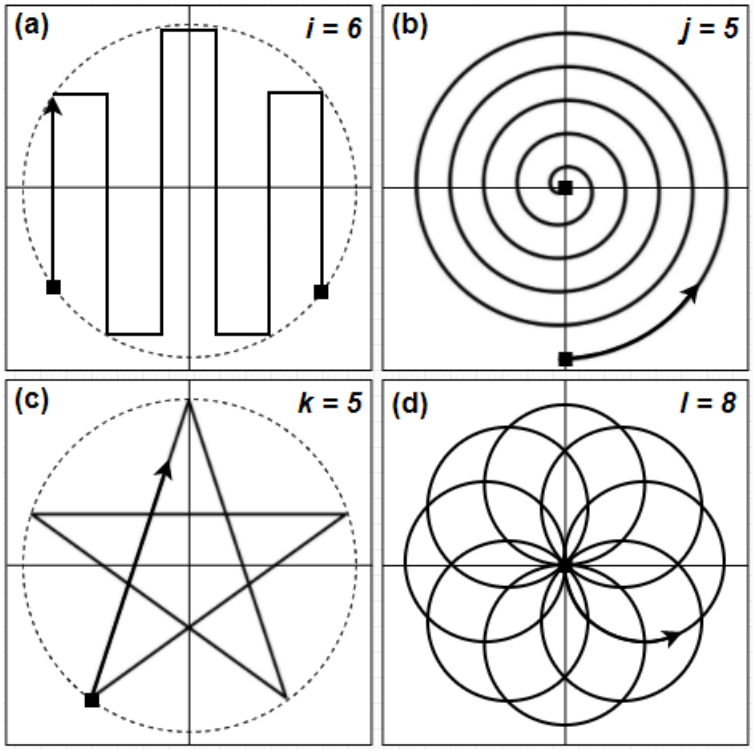
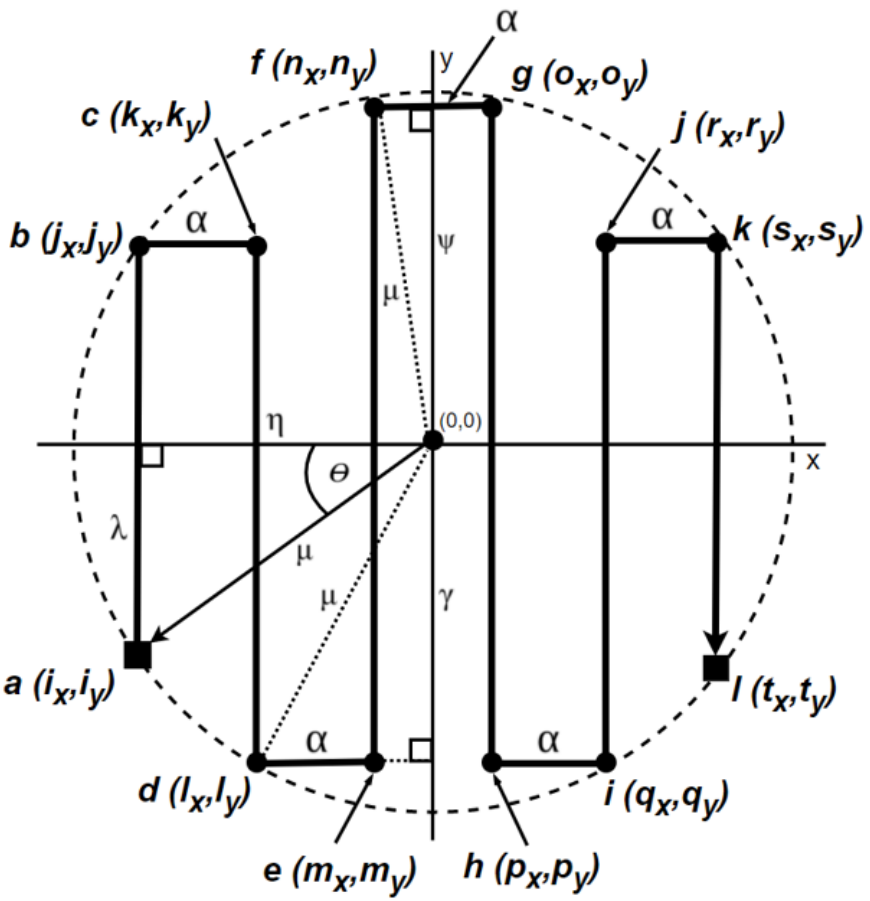


Figure 5.3: Abstract two-dimensional trajectory representations: **(a)** Lawn-mower, where number of passes $i = 6$, **(b)** Spiral, where number of rotations $j = 5$, **(c)** Star, where number of points $k = 5$, and **(d)** Flower, where number of petals $l = 8$.

sembling that of approximate cellular decomposition trajectories; and (4) ***Flower*** inspired by sector search patterns and magnetic particle imaging techniques which provide a spatial emphasis but with the modification of the progressively angular distribution of sampling. Simulations and field experiments were conducted in open unobstructed virtual and physical environments. Thus, geometric patterns were sufficient to traverse the designated volume.

To drive the TAUS model within the simulation environment based on these four trajectories, we express them mathematically as functions of time to obtain constant speed while traversing through the bounded physical domain. The Spiral and Flower trajectories are written as continuous functions, while the Lawn-mower and Star are decomposed into piecewise continuous lines. The Euclidean-

Pythagorean mathematics behind the Lawn-mower, Archimedean Spiral, Star, and Flower trajectory can be seen in Figure 5.4, 5.5, 5.6, and 5.7. Each trajectory was pre-planned and carried out in a simulated environment (see Figure 5.8).



$\theta = 45^o$;

$\mu = 3.814$ m;

$\sigma = 5$; //number of passes -1

$\alpha = \frac{2 \cdot i_x}{\sigma}$;

***a*(i_x,i_y)**:
$i_x = \eta$;
...where $\eta = (\cos(\theta) \cdot \mu) \cdot -1$;
$i_y = \lambda$;
...where $\lambda = (\sin(\theta) \cdot \mu) \cdot -1$;

***b*(j_x,j_y)**:
$j_x = i_x$;
$j_y = |i_y|$;

***c*(k_x,k_y)**:
$k_x = j_x + \alpha$;
$k_y = j_y$;

***d*(l_x,l_y)**:
$l_x = k_x$;
$l_y = \gamma$;
...where $\gamma = (\sqrt{\mu^2 - l_x^2}) \cdot -1$;

***e*(m_x,m_y)**:
$m_x = l_x + \alpha$;
$m_y = l_y$;

***f*(n_x,n_y)**:
$n_x = m_x$
$n_y = \psi$;
...where $\psi = \sqrt{\mu^2 - n_x^2}$;

***g*(o_x,o_y)**:
$o_x = n_x + \alpha$;
$o_y = n_y$;

***h*(p_x,p_y)**:
$p_x = o_x$;
$p_y = m_y$;

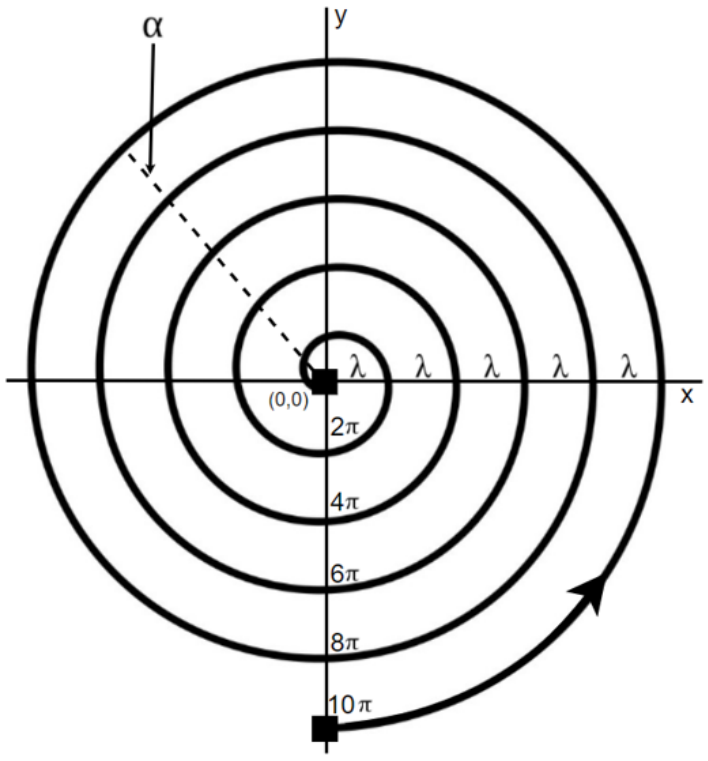
***i*(q_x,q_y)**:
$q_x = p_x + \alpha$;
$q_y = p_y$;

***j*(r_x,r_y)**:
$r_x = q_x$;
$r_y = k_y$;

***k*(s_x,s_y)**:
$s_x = r_x + \alpha$;
$s_y = r_y$;

***l*(t_x,t_y)**:
$t_x = s_x$;
$t_y = i_y$;

Figure 5.4: The Lawn-mower trajectory Euclidean-Pythagorean mathematics that drives the simulations of the TAUS model.



$\alpha = 3.814$ [m];

$v = 0.5$ $\left[\frac{m}{s}\right]$;

$\phi = 5$; //number of revolutions;

$\beta = \frac{\alpha}{\phi} = 0.7628$ [m];

$\eta = \frac{\beta}{2 \cdot \pi} = 0.1214$ $\left[\frac{m}{radian}\right]$;

$r = \eta \cdot \theta$;
...where θ is an angular variable;

$\rho = \sqrt{2} \cdot \sqrt{\frac{0.5 \cdot t}{\eta}}$

t = simulation monotonic time value [s];

$v \cdot \tau = \delta \therefore \tau = \frac{\delta}{v}$ [s]; //Newtonian kinematics
...where δ = distance [m], and τ = time [s];

Polar Arc Length:

$L_{arc} = \int_a^b \sqrt{(r)^2 + (\frac{dr}{d\theta})^2}\, d\theta$;
...where $a = 0$;
...where $b = 2\pi, 4\pi, 6\pi, 8\pi$, and 10π;
...where $\frac{dr}{d\theta} = \eta$;

$L^1_{arc} = \int_0^{2\pi} \sqrt{(\eta \cdot \theta)^2 + (\frac{d}{d\theta}(\eta \cdot \theta))^2}\, d\theta = 2.530$ [m];
...where $\tau^1 = \frac{L^1_{arc}}{v} = 5.059$ [s];

$L^2_{arc} = \int_{2^\pi}^{4\pi} \sqrt{(\eta \cdot \theta)^2 + (\frac{d}{d\theta}(\eta \cdot \theta))^2}\, d\theta = 7.088$ [m];
...where $\tau^2 = \frac{L^2_{arc}}{v} = 14.176$ [s];

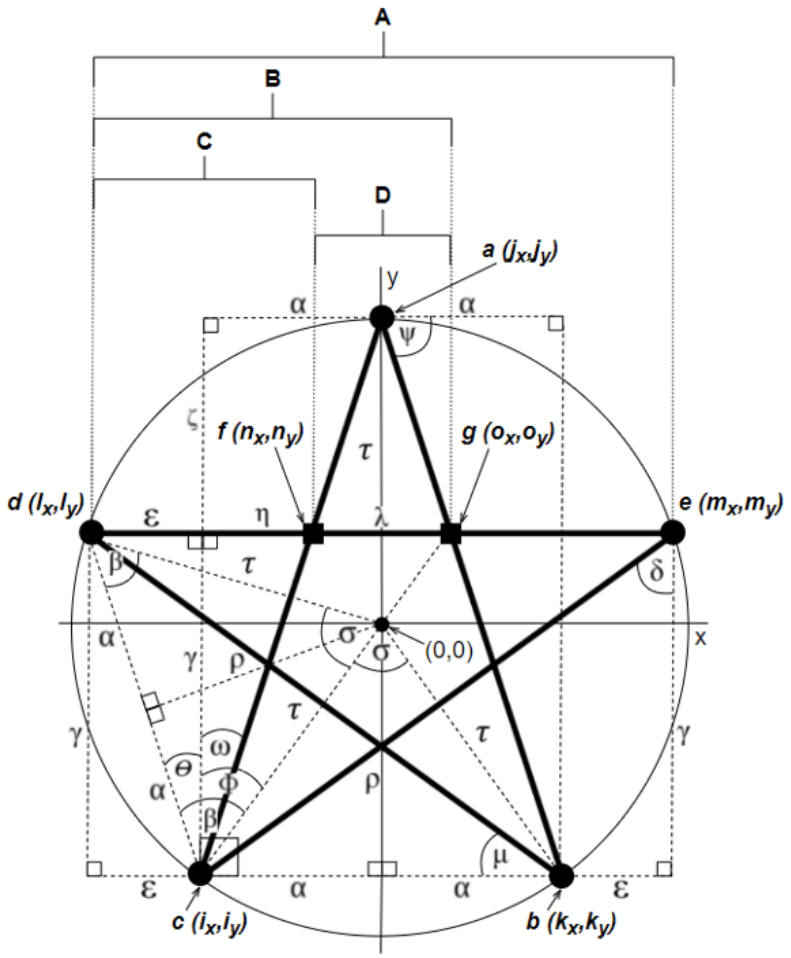
$L^3_{arc} = \int_{4^\pi}^{6\pi} \sqrt{(\eta \cdot \theta)^2 + (\frac{d}{d\theta}(\eta \cdot \theta))^2}\, d\theta = 11.769$ [m];
...where $\tau^3 = \frac{L^3_{arc}}{v} = 23.540$ [s];

$L^4_{arc} = \int_{6^\pi}^{8\pi} \sqrt{(\eta \cdot \theta)^2 + (\frac{d}{d\theta}(\eta \cdot \theta))^2}\, d\theta = 16.459$ [m];
...where $\tau^4 = \frac{L^4_{arc}}{v} = 32.919$ [s];

$L^5_{arc} = \int_{8^\pi}^{10\pi} \sqrt{(\eta \cdot \theta)^2 + (\frac{d}{d\theta}(\eta \cdot \theta))^2}\, d\theta = 21.154$ [m];
...where $\tau^5 = \frac{L^5_{arc}}{v} = 42.308$ [s];

$\sum_{n=1}^{5} \tau^n = 188.36$ [s];

Figure 5.5: The Archimedean Spiral trajectory Euclidean-Pythagorean mathematics that drives the simulations of the TAUS model.



τ = radius;

$\sigma = \frac{1}{5} \cdot 2\pi$;

$\beta = |\pi - (\frac{1}{2} \cdot \sigma + \frac{\pi}{2})|$;

$\theta = |2 \cdot \beta - \frac{\pi}{2}|$;

$\zeta = |j_y - l_y|$;

$\omega = \tan^{-1}(\frac{\alpha}{\gamma+\zeta})$;

$\phi = |\beta - \theta|$;

$\psi = \tan^{-1}(\frac{\zeta+\gamma}{\alpha})$;

$\mu = \tan^{-1}(\frac{\gamma}{\epsilon+2\cdot\alpha})$;

$\delta = \tan^{-1}(\frac{2\cdot\alpha+\epsilon}{\gamma})$;

A = $|m_x - l_x|$;
B = $|o_x - l_x|$;
C = $|n_x - l_x|$;
D = $|o_x - n_x|$;

Extreme and Mean Ratio:
(Golden Ratio)

$\frac{A}{B} \approx \frac{B}{C} \approx \frac{C}{D} \approx 1.618...$

***a*(j_x,j_y)**:
$j_x = 0$;
$j_y = \tau$;

***b*(k_x,k_y)**:
$k_x = \alpha$;
...where $\alpha = \sin(\frac{1}{2} \cdot \sigma) \cdot \tau$
k_y = -ρ;
...where $\rho = \cos(\frac{1}{2} \cdot \sigma) \cdot \tau$

***c*(i_x,i_y)**:
i_x = -k_x;
$i_y = k_y$;

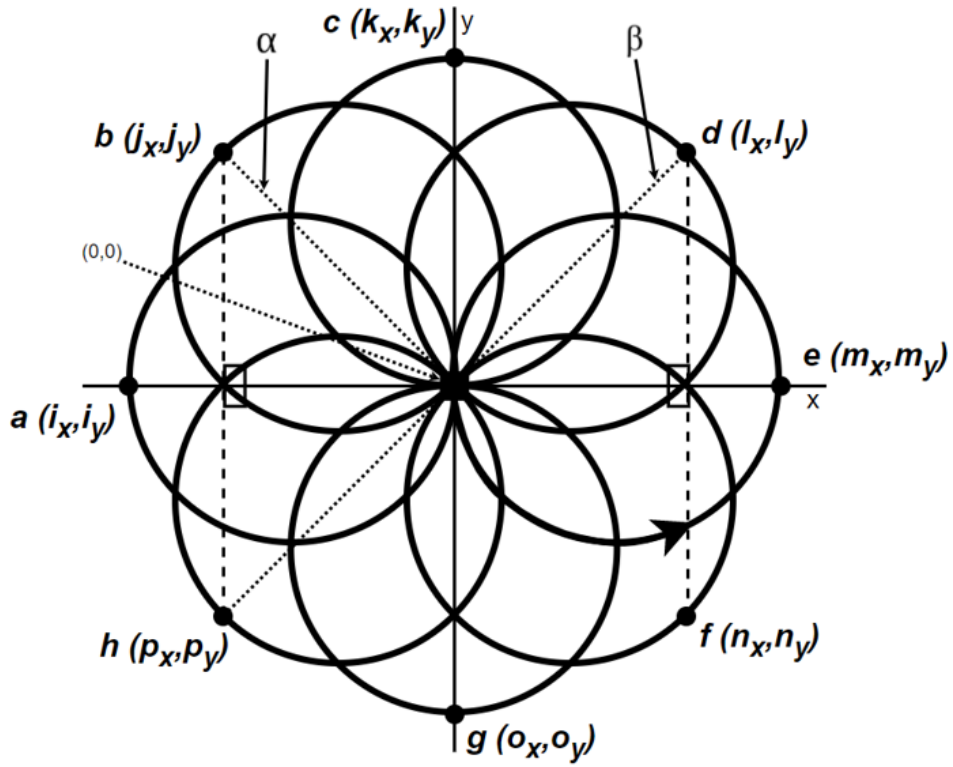
***d*(l_x,l_y)**:
l_x = -$|\epsilon + \alpha|$;
...where $\epsilon = \sin(\theta) \cdot 2\alpha$
$l_y = |i_y$ - $\gamma|$;
...where $\gamma = \cos(\theta) \cdot 2\alpha$

***e*(m_x,m_y)**:
$m_x = |l_x|$;
$m_y = l_y$;

***f*(n_x,n_y)**:
$n_x = \epsilon + \eta$;
...where $\eta = \tan(\omega) \cdot \gamma$
$n_y = l_y$;

***g*(o_x,o_y)**:
$o_x = n_x + \lambda$;
...where $\lambda = (tan(\phi) \cdot \gamma) - \eta$
$o_y = n_y$;

Figure 5.6: The pentagram (Star) trajectory Euclidean-Pythagorean mathematics that drives the simulations of the TAUS model.



$\alpha = 3.814$ m;

$\beta = 7.628$ m;

$v = 0.5\ \frac{m}{s}$;

$\tau = \frac{\delta}{v} = 23.96$ s;
...where $\delta = 2 \cdot \pi \cdot \frac{\alpha}{2}$;

$\eta = \frac{2 \cdot \pi}{\tau} = 0.2622\ \frac{rad}{sec}$;

***a*(i_x,i_y)**:
$i_x = \alpha \cdot -1$;
$i_y = 0$;

***b*(j_x,j_y)**:
$j_x = (\cos(45^o) \cdot \alpha) \cdot -1$;
$j_y = \sin(45^o) \cdot \alpha$;

***c*(k_x,k_y)**:
$k_x = 0$;
$k_y = \alpha$;

***d*(l_x,l_y)**:
$l_x = \cos(45^o) \cdot \alpha$;
$l_y = \sin(45^o) \cdot \alpha$;

***e*(m_x,m_y)**:
$m_x = \alpha$;
$m_y = 0$;

***f*(n_x,n_y)**:
$n_x = \cos(45^o) \cdot \alpha$;
$n_y = (\sin(45^o) \cdot \alpha) \cdot -1$;

***g*(o_x,o_y)**:
$o_x = 0$;
$o_y = \alpha \cdot -1$;

***h*(p_x,p_y)**:
$p_x = (\cos(45^o) \cdot \alpha) \cdot -1$;
$p_y = (\sin(45^o) \cdot \alpha) \cdot -1$;

Polar (r,θ) to Cartesian (x,y):
$y = \frac{\alpha}{2} \cdot \sin(\eta \cdot t - \lambda - \psi) \pm \phi$
$x = \frac{\alpha}{2} \cdot \cos(\eta \cdot t - \lambda - \psi) \pm \phi$
...where λ = t reset, ψ = radian increment, and ϕ = petal offset
...where $\frac{d}{dx}$ and $\frac{d}{dy}$ are velocity components at time t.

Figure 5.7: The Flower trajectory Euclidean-Pythagorean mathematics that drives the simulations of the TAUS model.

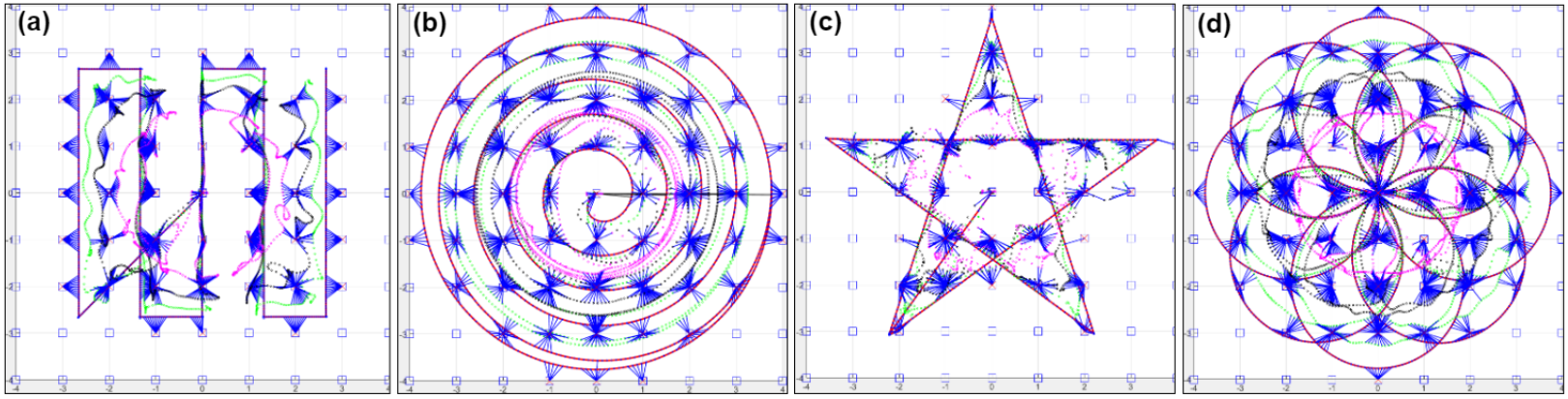


Figure 5.8: The simulation spatial data for the **(a)** lawn-mower, **(b)** spiral, **(c)** star, and **(d)** flower trajectories, respectfully. Spatial information for the sensors are the multi-colored dots, UAS model is the solid red line, and the blue lines are instances where sensor exposure was within distance requirements.

5.4 Factors for Evaluation

We identified factors that provide a spatiotemporal characterization of the TAUS that emphasize the impact of the proposed trajectories. We defined nine factors that represent several temporal, spatial, and sampling aspects of system performance: (1) flight duration, (2) volume encompassed, (3) coverage of discrete points in space, (4) coverage after 30 seconds of traversal (Cov-F), (5) remaining coverage with 30 seconds of traversal left (Cov-L), (6) dynamic temperature field reconstruction error, (7) detection time, (8) energy consumption, and (9) accumulated data.

The ***duration*** is simply the total time it takes the system to complete one full traversal of the bounded volume with a fixed velocity. The ***volume*** is calculated by leveraging a three-dimensional convex hull algorithm and providing as input the spatial points produced by each sensor along the tether. With respect to the ***coverage*** factor, we defined static sampling locations on a [1 m x 1 m x 1 m] grid throughout the bounded space. This provides a total number of viable sampling locations, N. We have also defined an arbitrary maximum sampling distance of 0.5 m from any given sampling location $s_i\{x_i,y_i,z_i\}$, where $i \in N$ when there is adequate sensor proximity exposure. The sampling rate is governed by the temperature sensor's intrinsic response time and user-configured output bit resolution. In order to determine adequate proximity between the sensors location $k_j\{x_j,y_j,z_j\}$ and the sampling location s_i the distance, δ, is defined as:

$$\delta(s_i, k_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{5.4}$$

Therefore, a $\delta \leq 0.5$ m indicates that the sensor array has successfully sampled the given spatial point s_i at least once. With a priori knowledge of N and their spatial information, we found the total number of distinctive points that were successfully

sampled, M, by invoking a Boolean operation each time-step while traversing the volume. Thus, we derived the percent coverage, P_{cov}, which is formally expressed as:

$$P_{cov} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} Bool(\delta(s_i, k_j))}{N} \tag{5.5}$$

Once we had P_{cov}, the coverage at particular time-steps throughout the simulation was produced. Specifically, the percent coverage at the 30-second time-step, ***Cov-F***, and the difference between the final P_{cov} and the time-step with 30 seconds remaining, ***Cov-L***, were produced. Since Cov-L requires P_{cov} it is a purely post-processed value. These factors provided us insight into how uniformly and temporally efficient each trajectory was for coverage.

The ability to sense a given volume is determined by the accuracy of the sensed data representing the true state of the atmospheric system at any given time and location. Thus, during traversal, we ***reconstructed*** the temperature field from the sensor spatial and sampled temperature information each time-step. We provided the simulation and modeled sensors either a known static or dynamic temperature field gradient to traverse through. Input temperatures for the static field represented by a [8 m x 8 m x 15 m] bounded three-dimensional grid ranged from an arbitrary 1.75 °C to 3.75 °C with a 0.250 °C ambient temperature step resolution per 1 m translation. A dynamic gradient allowed us to emulate a cold front moving through the volume at a fixed rate as the model traversed the grid. For dynamic fields the entire volume was initialized to 3.50 °C and as the simulation iterated the cold-front translated across the volume at a fixed 0.250 m step. Grid cell values were decremented by 0.100 °C each step. After each time-step, the temperature-field of the bounded volume was reconstructed using

natural-neighbor interpolation for smooth approximations. This allowed us to evaluate the sensor model array accuracy under each trajectories spatial influence via root-mean-square-error (RMSE) analysis to the known input field values at each time step.

Once any of the sensors along the array ***detected*** a deviation from the initial temperature, the time stamp was recorded and turned into a percentage of the trajectory's total duration. This detection factor told us how temporally effective a given trajectory was at interpreting the state of the input field. To account for inadvertent directional bias in dynamic temperature field reconstruction performance, we ran a series of four simulations for each trajectory where the cold-front translated from a different cardinal direction. We then took the average of the reconstruction error and detection times to reduce the potential impact of this bias.

Energy consumption was based on the total amount of energy in kilo-joules [kJ] the system used to traverse the volume assuming an average power consumption of 120 Watts (12 volts·10 amps) for un-tethered UAS, and 360 Watts (24 volts·15 amps) for tethered UAS. These are empirical values derived from physical system data. The reason for such a discrepancy between un-tethered and tethered UAS is the innate inefficiencies with power step-up/step-down transformation and physical media transmission losses across lengthy high gauge conductors.

Accumulated ***data*** was taken as the total number of net sensor data packets [pkt] transmitted at 0.42 Hz, which is the result of the difference in raw and erroneous data. We expected the performance of the sensor array to scale linearly with the number of sensors in the array.

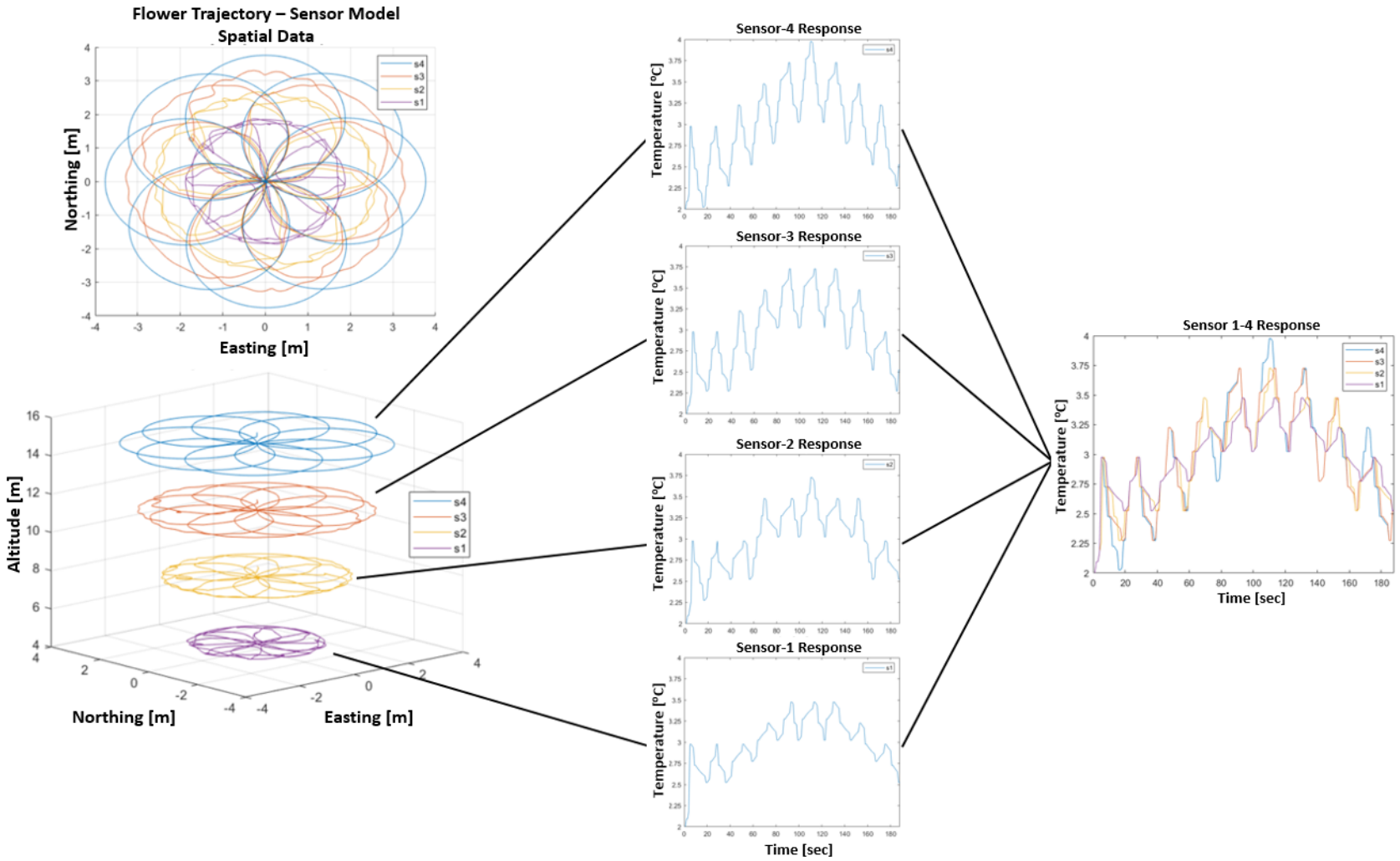


Figure 5.9: The Flower trajectory is shown from multiple angles along with the individual sensors temperature response to a static input temperature field.

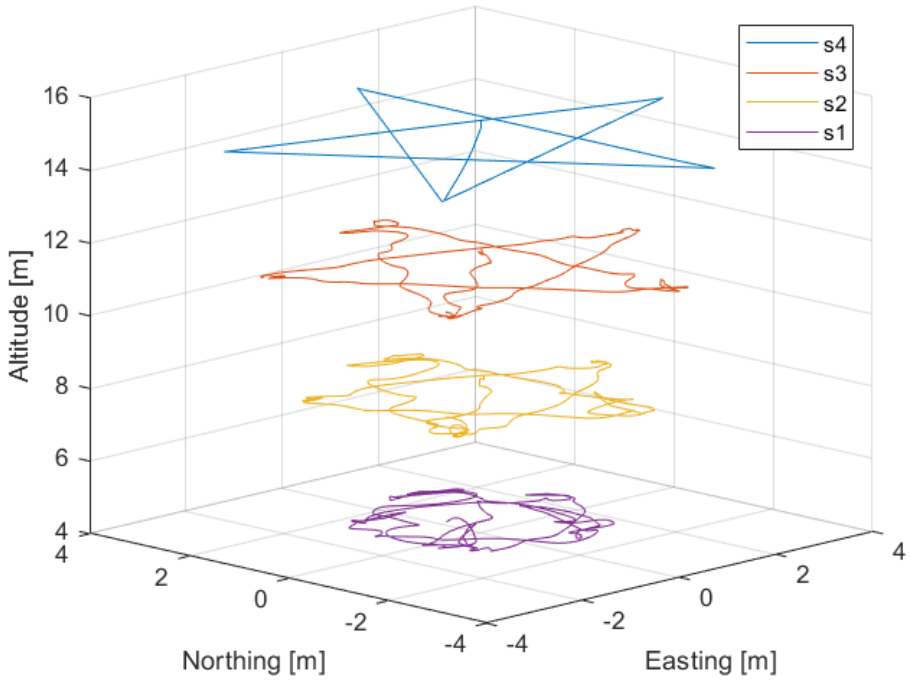


Figure 5.10: The sensor array spatial data related to a single tethered Star traversal by the TAUS simulation. Sensor-4 (s4) is located on-board the UAS while Sensor-1 (s1) is located lowest near the ground station.

In this section, we describe the simulation setup and temperature field reconstruction. We use a physics-based simulation of tethered and un-tethered UAS models for evaluation. We analyze the influence of the trajectories on the nine factors and evaluate overall performance.

5.5 Temperature Field Reconstruction

We performed eight simulations, where four used a tethered UAS model with an array of four modeled sensors equally spaced (see Figure 5.9 and 5.10), and four were for an un-tethered UAS model with a single modeled sensor integrated on-board as a baseline to compare against. Each type of trajectory was simulated using either model type for a single traversal. It is assumed that performance scales linearly with traversal iterations without modification to the simulation. Despite

the type of trajectory, the rate at which the UAS traversed was fixed at $0.5\,\mathrm{m/s}$ to standardize the comparison and provide constant aspiration to the sensors.

To drive the simulation with each trajectory's mathematical formulation, we referenced a trajectory provider function for each fixed monotonic time-step of 0.001 seconds. For each time-step t, the simulator takes as input a 7-tuple that comprises of 3-positions, 3-velocities, and a yaw reference for the UAS to follow. We, therefore, write time-parameterized forms for each of our trajectories, such that can be written as a Matlab function:

$$[\alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, \phi] = \text{trajectory\_reference(t)} \tag{5.6}$$

where α, β, γ denote the northing, easting, and z-down positional components respectively, and $\dot{\alpha}$, $\dot{\beta}$, $\dot{\gamma}$ are the first derivative producing the respective velocity components. The final position of the tuple represents the model's yaw (ϕ) about the z-axis controlled from $\pm\pi$. The simulator employs a standard feedback linear-quadratic-regulator (LQR) controller that generates the attitude and thrust commands necessary to follow the reference in the simulated environment dictated by Newtonian kinematics.

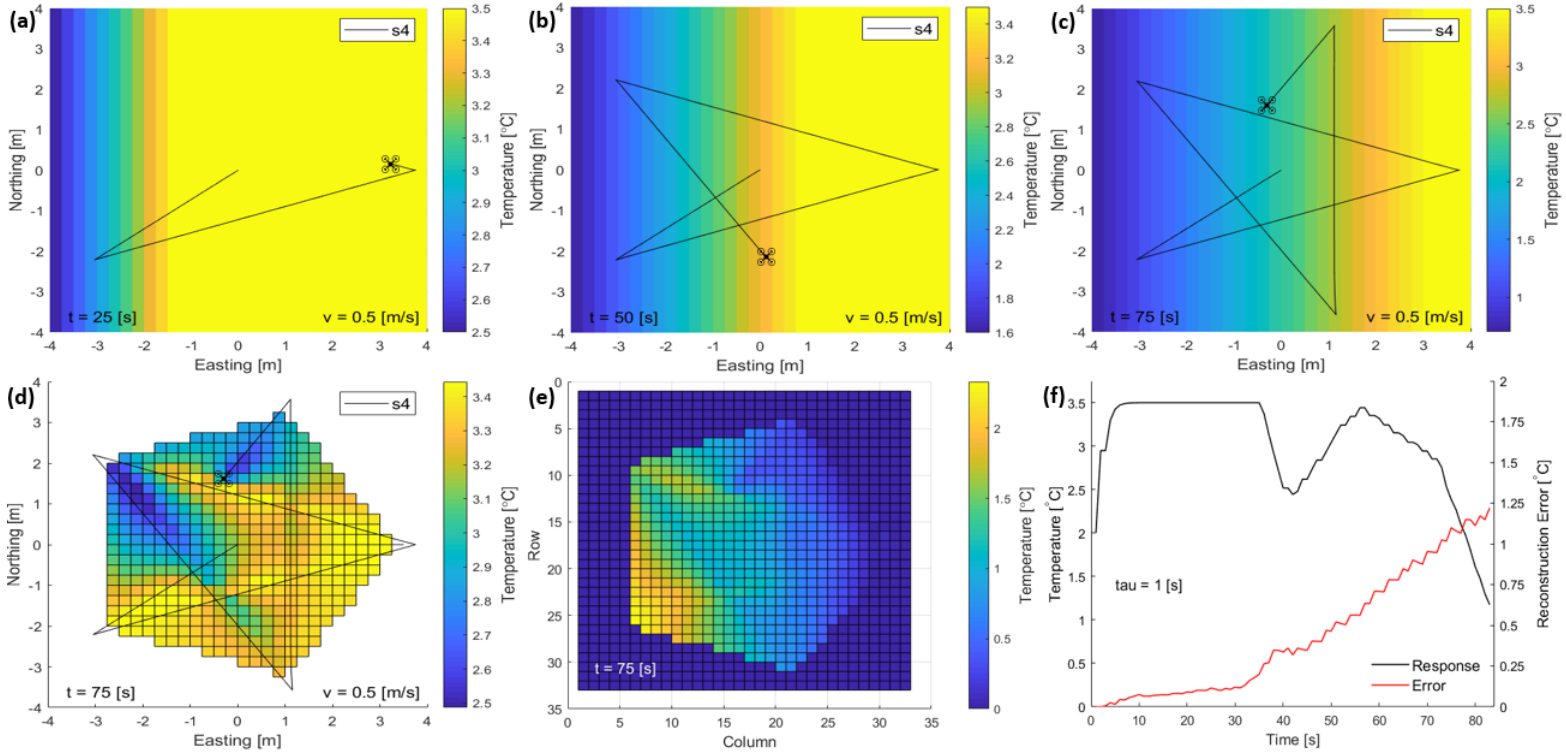


Figure 5.11: **(a,b,c)** The dynamic input temperature field resembling a cold-front tracking east with Star trajectory positional information for sensor-4 (s4) at time t = 25 seconds, 50 seconds, and 75 seconds, respectively. A UAS symbol is presented to emphasize relative position in time. **(d)** Temperature field reconstruction via natural-neighbor interpolation. **(e)** Absolute difference between input and reconstructed data in space. **(f)** The response of s4 and the reconstruction error through time.

We ran each trajectory in the presence of both static and dynamic input temperature fields in the form of a cold-front and subsequently reconstructed the traversed field and determined error at each time step. The static input fields do not have a time-varying aspect so they did not apply to the detection factor. Static horizontal temperature fields do not naturally occur in our planet's troposphere so we focused on the dynamic input fields and reconstructions. Figure 5.11 shows a dynamic field translating across the domain as the model ran a Star trajectory. The field translates as a function of the trajectories duration at increments that allowed for 32 steps at 0.250 m each. Therefore, the rate the dynamic cold front translated through the volume was between 0.042 and 0.094 m/s. The TAUS is designed to traverse laterally, thus, we provide lateral cold-fronts. Each sensor sits in a vertical band throughout the traversal so any gradient the sensors would quickly asymptote toward a perceptible steady-state independent of trajectory influence. The reconstruction error is highly dependent on the intrinsic τ value for the sensors. This value dictates how quickly the sensors can react to a changing temperature field, so we set all sensor models $\tau = 1$ second, a value typical of physical high-end ambient temperature sensors. The rate of traversal impacts the time each sensor gets exposed to certain temperature fields. Therefore, increased speeds can reduce the sensor's ability to reach the incident temperature resulting in reduced variance and overall reconstruction performance. This problem is comparable to that of path following which have been solved by multi-dimensional Carrot-Chasing algorithms that look for asymptotic convergence with a changing target value [70]. The τ coupled with traversal speed dictates how close the follower, in this case, the ambient temperature sensor, can sample the changing reference, a dynamic temperature front.

5.5.1 Analysis

The outcomes of the factor evaluations are presented in Table 5.1 along with the mean and median statistics in the final two rows. To determine the percentage of the overall volume each trajectory encompassed, Equation (5.2) was evaluated with $h = 14.5$ to produce 330.260 m^3 of maximum viable volume for traversal. Since the static number of sampling points s_i are distributed in a uniform 3-dimensional grid the total number of reachable sampling points is taken as $\lfloor V_{par} \rfloor = 330$.

The un-tethered simulations were significantly outperformed by the tethered simulations with respect to volume, coverage, Cov-F, Cov-L, and data, regardless of the trajectory. On average the tethered trajectories sampled a volume 34.49 times greater, generated a P_{cov} 2.96 times larger than un-tethered trajectories for the same duration of the flight. This is an expected outcome because the tethered trajectories take advantage of the sensor array throughput. We suspect if the untethered trajectories were allowed to make more than one pass and descend in the vertical dimension they would have more competitive factor performance, specifically with respect to volume and coverage. The negative trade-off would be the increased flight duration and energy consumption being a multiplicative factor equal to the number of passes. Additionally, the un-tethered system would not be able to simultaneously sample the areas it descended from introducing room for increased reconstruction error through more expansive interpolation.

Table 5.1: The simulated trajectory outputs for a single traversal either un-tethered (U) or tethered (T) for the factors duration, volume, percent coverage, percent coverage in the first 30 seconds, percent coverage with 30 seconds remaining, temperature reconstruction error, detection time, energy consumed, and data throughput, respectively. The outputs are derived from the system performance evaluation factors defined in Section III. *Top* performance per factor is in **bold**.

Trajectory	Dur. [s]	Vol. [m^3]	Cov. [%]	Cov-F [%]	Cov-L [%]	Err. [°C]	Det. [%]	En. [kJ]	Data [pkt]
Mower-U	89.24	7.83	9.39	3.03	2.72	1.21	35.01	10.70	37.48
Mower-T	89.24	274.80	32.72	11.51	**9.69**	1.23	39.35	32.12	149.92
Spiral-U	165.29	8.70	17.27	3.63	1.51	1.35	31.00	19.83	69.42
Spiral-T	165.29	**291.34**	**43.93**	**13.93**	2.72	1.32	32.40	59.50	277.70
Star-U	**84.82**	6.76	8.78	3.93	2.42	1.17	39.19	**10.17**	35.62
Star-T	**84.82**	238.70	29.39	12.72	8.78	**1.16**	39.93	30.53	142.50
Flower-U	188.48	8.40	13.63	4.24	0.30	1.18	**27.58**	22.61	79.16
Flower-T	188.48	289.14	39.39	13.63	1.21	1.19	34.95	67.85	**316.64**
Mean-U	131.96	7.92	12.27	3.71	1.74	1.23	33.20	15.83	55.42
Mean-T	131.96	273.50	36.36	12.95	5.60	1.23	36.66	47.50	221.69
Median-U	127.27	8.12	11.51	3.78	1.97	1.19	33.01	15.27	53.45
Median-T	127.27	281.97	36.06	13.18	5.75	1.21	37.15	45.81	213.81

We anticipated the tethered simulations would have a noticeable advantage with respect to reconstruction error and detection, but we see comparable results. This is due to the vertically uniform dynamic input temperature fields. If there was a vertical gradient where colder temperatures entered the traversable volume low to high while tracking laterally, better emulating the thermodynamics of a cold-front where warm air rises, we would see un-tethered performance decrease dramatically for both factors. Tethered system sensor position quality degrades with sensor locations lower on the tether due to the elastic nature of the slacked cable crossing over the origin where slack would be at the maximum.

The Mower trajectory both tethered and untethered showed the most agreement between the Cov-F and Cov-L factors. This indicates that the traversal coverage was most spatiotemporally uniform. Conversely, the Flower trajectory presented major disagreement indicating this trajectory increasingly re-samples locations throughout the traversal due to its natural cyclical overlap. The tethered Spiral trajectory maximizes volume, coverage, and Cov-F. Nevertheless, it should encompass nearly the entire available bounded volume but the simulation only reported $291.349\,\mathrm{m}^3$ or 88.29% of the rated volume, therefore $\sim$11.71% is unaccounted for. We suspect the majority of that missing percentage is due to the rigid construction of the three-dimensional convex hull, and the simplified quadratic expression (s^2) used to describe the catenary behavior of the tether. The quickest trajectory was the Star, which also produced the lowest reconstruction error and energy consumption. Since the Star samples quickly across the traversable volume in all major cardinal directions it has the opportunity to generate sample points that have been influenced by the dynamic input field throughout the traversal.

Chapter 6

Discussion & Conclusions

This chapter focuses on a discussion of the overall work presented in prior chapters and inferences derived from the outcomes in section 6.1. The works limitations and assumptions are presented in section 6.2 and concluding remarks can be found in section 6.3. The future direction of this research with details for a set of specific research questions are described in section 6.4.

6.1 Discussion

This section summarizes the results from Chapters 3, 4, and 5 and how they attempt to answer the research questions 1-4 posed in Chapter 1, which are restated here:

1. **Can we develop a proof-of-concept UAS that can increase flight times and data throughput (without profiling) for the agricultural and atmospheric sectors?**

2. **Will the power transmission efficiency bias the atmospheric temperature sensors?**

3. **Can we validate the systems base-level operational performance in the field?**

4. **Since the UAS will be tethered and therefore bounded creating a new system limitation, can we evaluate a set of system trajectories against well defined metrics to determine operational performance?**

Chapter 3 was written to answer research question 1 by developing the TAUS. Specifically, Chapter 3 discovered that we can leverage power-over-tether UAS technology to increase UAS flight times by engineering prototype TAUS-Alpha, a desktop small-scale testbed. This warranted the development of TAUS-Beta, a full sized system with a low-quality temperature sensor array integrated along the tether for proof-of-concept testing. TAUS-Beta was lab tested and showed promise as a viable system to be deployed in the field for further testing. We took the system to the field and demonstrated a 6-hour continuous quasi-static 15m AGL flight.

Chapter 4 was written to answer research questions 2 and 3. To address question 2, a temperature sensor array was integrated in-line and off-line to test whether or not the power transmission efficiency of the system could bias the sensors. The system and modified sensing-power-tether was taken to the field and tested for 1-hour of continuous quasi-static flight at sunset and 15m AGL which found that adequate power transmission efficiency is a function of the conductor bottle-neck, which is the current rating of the traces on the sensor PCB. If the rating is approached the electrons will generated radiant heat due to quantum collisions across the conductors. With that finding, the TAUS-Beta with an adequately efficient power-tether transmission and non-biased sensor configured array was taken to RMF to be tested for validation through intercomparison analysis to address research question 3. Specifically, the system was tested for 77 minutes of continuous quasi-static flight at sunrise and 15m AGL which found minimal

sensing error for sensors on the TAUS and the RMF tower at comparable AGL validating the system's base operation.

Chapter 5 was written to answer research question 4 by developing a representative TAUS computer model and simulating it in a physics emulated environment while traversing via an optimal controller. This allowed us to evaluate a set of bounded trajectories for the TAUS model against well defined factors. We found that the TAUS outperformed un-tethered systems for the majority of factors. Specifically, the tethered Star trajectory minimized performance factors related to reconstruction error (1.169 °C), duration (84.825 seconds), and energy consumption (10.179 kJ).

6.2 Assumptions & Limitations

As with any research project, there were several assumptions made and limitations determined or identified throughout this work. We acknowledge this reality and present a non-exhaustive list with brief explanation:

1. Sensors integrated along the power-tether in an array configuration was assumed to be adequately aspirated. A temperature sensors response time is highly dependent on the rate of air flow it is exposed to. We considered this a reasonable assumption because the sensor array will experience varying levels of wind and it is constantly displaced through the atmosphere at a fixed rate due to the system traversal.

2. The off-tether sensor array placement of 4 inches was assumed to be adequate distance to isolate potential bias from the power-tether thermal radiation due to power transmission inefficiencies. This was considered a reasonable

distance because the thermal radiation is limited to minimal warming sensed through conduction not convection.

3. The meteorological tower at RMF and it's temperature sensors at 2m and 10m AGL are assumed to be relatively high-quality, well maintained, and appropriately calibrated. We consider this a reasonable assumption because technicians actively service, maintain, and achieve data from the tower.

4. In simulation, we assume that the sensors along the array are ideal. This means the temperature response time is fixed and reliable. We consider this a reasonable assumption because physical temperature sensors are designed to be invariant, but we acknowledge perturbations and inappropriate handling of the device could alter the sensors reliability, specifically a bead thermistor.

5. We arbitrarily assumed a sensor distance of 0.5m to any given position in space on our sampling grid was to be considered successfully sampled. This was purely a spatial consideration where even short exposure times were assumed successful.

6. The average power consumed by the system is assumed to not depend on the trajectory selected. This was considered a reasonable assumption because the system was fixed to 0.5 m/s traversal speed independent of the selection. This was done to help uniform the trajectories metric analysis. In reality, the power demands will be different because some trajectories are continuous functions and do not experience start-stop accelerations. We assume this to be negligible even if it was taken into account.

7. We chose to approximate the power-sensing-tether's physical position at maximum spatial extent to be represented as a parabolic function instead of

a hyperbolic cosine function, or catenary. We assumed this was a reasonable abstraction because we were dealing with relatively small values of distance. A hyperbolic cosine function will grow exponentially whereas a parabolic will grow in a quadratic manner, thus their differentiation will become more prominent at larger values.

8. The four trajectories identified and used for simulations are assumed to be independent and to provide spatially unique traversal information warranting performance evaluation against the defined metrics.

9. The TAUS battery bank capacity of 66 Ah without photovoltaic recharge provides a maximum continuous flight time limit of $\sim$8 hours for a DJI F450 quad-rotor powered by our system. Other system limitations are power-sensing-tether length and sensor array separation, which is $\sim$15m and $\sim$3.81m, respectively. These limits were determined based on typical ground truth tower heights, ideal atmospheric lapse rate, and x.

6.3 Conclusions

A novel UAS was proposed for applications in the agricultural sector. The TAUS was successfully prototyped and field tested, showing promise as a viable field instrument to help agricultural decision makers make more efficient and optimal choices. Field tests consisted of 6 consecutive hours of flight at 50 ft. AGL, and again at sunset for 1 hour to acquire temperature data. The empirical evaluation shows system robustness and an ability for the system to be redeployed. Our experiments have shown some of the limitations of the system and indicated directions for improvement. There is a significant lack of system performance monitoring, and autonomous abilities preventing this device from being an isolated

field instrument prepared for self-deployment. There needs to be more of a quantitative and statistical analysis of the sensor data against high-quality and trusted AGL temperature sensors, perhaps those integrated along an EC tower. Integrating other IC's to sample data such as pressure, relative humidity, and volatile organic compounds (VOCs) will enhance the usefulness of the system to agricultural decision makers. Lastly, creating a network of these systems would increase their overall spatial resolution.

We conducted a 1-hour static field experiment with the TAUS against a ground truth tower installation we found minimal temperature sensing error of 0.143 °C and 0.091 °C between sensors at comparable altitude. Using the information acquired from the experiments, we developed a model that represents the physics that governs power-tethered UAS. We then ran simulations with the model in dynamic input temperature fields analyzing system performance factors that are influenced by our set of well-defined trajectories. We found that tethered systems outperformed un-tethered systems for the majority of factors. Specifically, the tethered Star trajectory minimized performance factors related to reconstruction error (1.169 °C), duration (84.825 seconds), and energy consumption (10.179 kJ).

In summary, the main contributions of this work are:

1. Development of a novel power-over-tether UAS prototype that includes sensors placed along the tether to provide high spatiotemporal resolution sampling of atmospheric temperature.

2. Field experiments and results demonstrating validation of operational performance of this platform to sense temperature fluctuations within a 15m above ground level range.

3. Development of a model that represents the physics governing power-over-tether UASs in flight.

4. Design and implementation of power-over-tether trajectories for the model to evaluate theoretical performance via factors in a bounded volume of a simulated atmosphere.

6.4 Future Work

Our presented approach can be improved by incorporating machine learning techniques to discover imperceptible trajectories that optimize various metrics. We plan to better differentiate tethered and un-tethered system performance via field reconstruction error when there is a more representative cold-front with a transient vertical aspect. We are currently working on a sensor feedback pipeline that would effectively close-the-loop for the system. This will allow us to make trajectory control and sensing advances which will enable autonomous field deployments. We will carry out a set of trajectory-based field experiments with the TAUS to validate our simulations theoretical outcomes. To maximize the effectiveness of the TAUS, we will frame our work as an optimization problem leveraging Markov decision processes to create a more comprehensive optimal policy. This will allow us to develop control and trajectory selection algorithms to be integrated into the physical system to perform online optimizations of the trajectory based on the sensed data. With these advances, we will produce a journal paper that will include our proceedings papers to ASABE [20] and IROS [21] while highlighting the aforementioned future work advancements. A potential candidate journal we have identified is the Journal of Environmental Modelling & Software [71] because it showcases research at the nexus of computer science and fields such as biological,

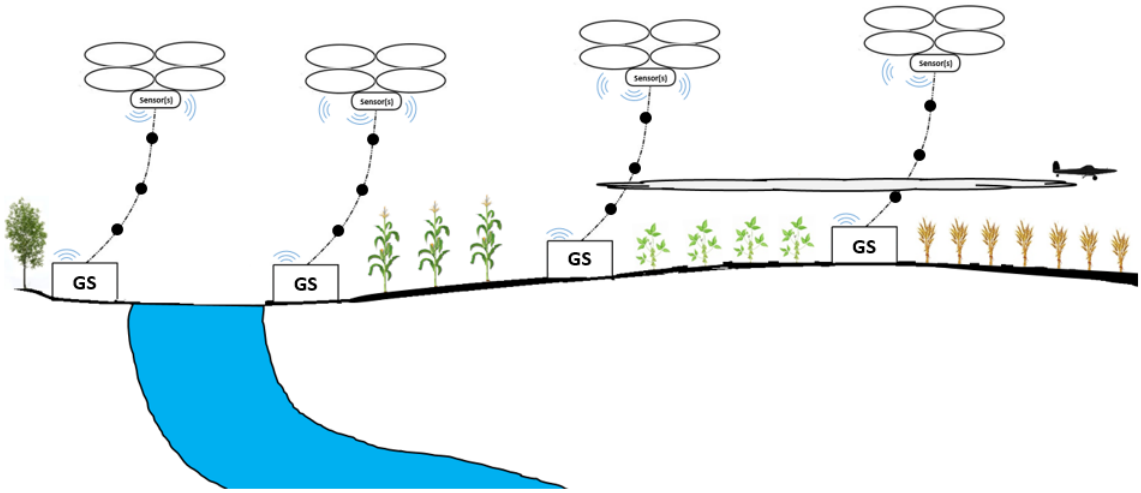


Figure 6.1: A high-level concept diagram of the future TAUS farm-to-farm network.

agricultural, and environmental engineering.

Looking forward beyond the immediate remaining tasks for a journal paper, there are many TAUS derivative research questions we can work to answer. At the start of my Ph.D. we intend on addressing a number of these before transitioning into a more challenging computer science and field robotics area of research.

1. Can we develop and deploy a farm-to-farm network of n-TAUS nodes? Since the start of this thesis we identified this as an interesting path forward (see Figure 6.1). The potential cooperation between nodes could expand the high spatiotemporal resolution impact.

2. Can we couple a TAUS with a center-pivot or linear traversing irrigation systems via rail system that spans the entire run which would be powered off the same electrical supply? This mechatronic mutualism would expand a TAUS spatial impact and in parallel increase our precision agriculture.

3. Similarly, could be mobilize the GS of the TAUS making it an unmanned ground vehicle (UGV)? We could have the UAS and the UGV perform

collaborate tasks for environmental monitoring.

4. Can we couple the TAUS with an underground wireless sensor network (WSN), specifically soil moisture sensors since it was identified as the most important agricultural impact parameter by agricultural producers? Could we recharge an already established underground WSN, or even data mule for the WSN with the TAUS as it traverses? The TAUS could acquire above-ground atmospheric data while the underground soil moisture WSN gathers below-ground data. This sensor and device fusion could create a complete precision agriculture profile at high-spatiotemporal resolution.

5. Can we upgrade the sensor array from low-quality temperature sensors to a high-quality low-profile sensor suite which could include air quality sensors for CO_2 and volatile organic compounds (VOC) in addition to meteorological variable such as pressure, humidity, etc.? This could also include the incorporation of an imaging sensor suite on-board the UAS composed of high pixel resolution RGB, multi-spectral, thermal, and/or NDVI cameras to process crop phenotypes and increase the systems data throughput.

6. Can we *localize* the power-sensing tether while the system is deployed and traversing? This could help us better position the system for increased levels of sensing optimization. This might require that we incorporate an inertial measurement unit (IMUs) along with each sensor suite integrated in-line along the tether. Similarly, can we derive the wind's impact on the tether? Since the wind is the major source of system perturbation, deriving it's physical impact on the power-sensing-tether would help us better understand system performance in the wild.

7. The TAUS is a robotic system designed for field deployments. Since this environment is inherently dynamic and constraining, can we develop an *adaptive sampling* and control algorithm designed to inform the system on the when, where, and how of system operation and sampling?

8. If there was an on-board battery along with the power-tether and photovoltaic system recharge, can we quantify and algorithmically express power-aware *planning* and adaptive power use for operation of the system at different tether lengths and therefore AGL deployment? For instance, if the system AGL is $<$ 20m the UAS relies entirely on the power-over-tether and photovoltaic recharge, while 20m $\leq$ AGL $\leq$ 50m rely on a power mixture of all three, and any AGL $>$ 50m the on-board battery gets utilized exclusively. These would be algorithms to manage the power utilization to maximize system continuous flight-time and sensor throughput.

Bibliography

[1] S. Shekhar, J. Colletti, F. Muñoz-Arriola, L. Ramaswamy, and C. Krintz, "Intelligent Infrastructure for Smart Agriculture: An Integrated Food, Energy and Water System," p. 8. 1

[2] "NIFA FACT Workshop: High Throughput, Field-Based Phenotyping Technologies for the Genomes to Fields (G2F) Initiative - IOWA STATE UNIVERSITY." 1

[3] F. Lei and H. Yong, "Study on dynamic model of tractor system for automated navigation applications," *Journal of Zhejiang University-SCIENCE A*, vol. 6, no. 4, pp. 270–275, Apr. 2005. [Online]. Available: https://doi.org/10.1007/BF02842055 1

[4] J. Katupitiya, R. Eaton, A. Cole, C. Meyer, and G. Rodnay, "Automation of an Agricultural Tractor for Fruit Picking," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 3201–3206, iSSN: 1050-4729. 1

[5] K. G. Hubbard, N. J. Rosenberg, and D. C. Nielsen, "Automated Weather Data Network for Agriculture," *Journal of Water Resources Planning and Management*, vol. 109, no. 3, pp. 213–222, July 1983, publisher: American

Society of Civil Engineers. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9496%281983%29109%3A3%28213%29 1

[6] X. Dong, M. C. Vuran, and S. Irmak, "Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems," *Ad Hoc Networks*, vol. 11, no. 7, pp. 1975–1987, Sept. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870512001291 1

[7] A. Vit and G. Shani, "Comparing RGB-D Sensors for Close Range Outdoor Agricultural Phenotyping," *Sensors*, vol. 18, no. 12, p. 4413, Dec. 2018, number: 12 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/18/12/4413 1

[8] S. F. D. Gennaro, F. Rizza, F. W. Badeck, A. Berton, S. Delbono, B. Gioli, P. Toscano, A. Zaldei, and A. Matese, "UAV-based high-throughput phenotyping to discriminate barley vigour with visible and near-infrared vegetation indices," *International Journal of Remote Sensing*, vol. 39, no. 15-16, pp. 5330–5344, Aug. 2018, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01431161.2017.1395974. [Online]. Available: https://doi.org/10.1080/01431161.2017.1395974 1

[9] T. Arnold, M. De Biasio, A. Fritz, and R. Leitner, "UAV-based multispectral environmental monitoring," in *2010 IEEE SENSORS*, Nov. 2010, pp. 995–998, iSSN: 1930-0395. 1

[10] A. K. Saha, J. Saha, R. Ray, S. Sircar, S. Dutta, S. P. Chattopadhyay, and H. N. Saha, "IOT-based drone for improvement of crop quality in agricultural field,"

in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2018, pp. 612–615. 1

[11] K. Uto, H. Seki, G. Saito, and Y. Kosugi, "Characterization of Rice Paddies by a UAV-Mounted Miniature Hyperspectral Sensor System," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 851–860, Apr. 2013, conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 1

[12] A. K. Saha, J. Saha, R. Ray, S. Sircar, S. Dutta, S. P. Chattopadhyay, and H. N. Saha, "IOT-based drone for improvement of crop quality in agricultural field," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2018, pp. 612–615. 1

[13] F. G. Costa, J. Ueyama, T. Braun, G. Pessin, F. S. Osório, and P. A. Vargas, "The use of unmanned aerial vehicles and wireless sensor network in agricultural applications," in *2012 IEEE International Geoscience and Remote Sensing Symposium*, July 2012, pp. 5045–5048, iSSN: 2153-7003. 1

[14] J. Wivou, L. Udawatta, A. Alshehhi, E. Alzaabi, A. Albeloshi, and S. Alfalasi, "Air quality monitoring for sustainable systems via drone based technology," in *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, Dec. 2016, pp. 1–5, iSSN: 2151-1810. 1

[15] D. Anthony, S. Elbaum, A. Lorenz, and C. Detweiler, "On crop height estimation with UAVs," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2014, pp. 4805–4812, iSSN: 2153-0866. 1

[16] U. R. Mogili and B. B. V. L. Deepak, "Review on Application of Drone Systems in Precision Agriculture," *Procedia Computer Science*, vol. 133, pp. 502–509, Jan. 2018. 1

[17] Y. A. Pederi and H. S. Cheporniuk, "Unmanned Aerial Vehicles and new technological methods of monitoring and crop protection in precision agriculture," in *2015 IEEE International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)*, Oct. 2015, pp. 298–301. 1

[18] M. E. Lussier, J. M. Bradley, and C. Detweiler, "Extending Endurance of Multicopters: The Current State-of-the-Art," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2019-1790. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2019-1790 1

[19] G. B. Frisvold and A. Murugesan, "Use of Weather Information for Agricultural Decision Making," *Weather, Climate, and Society*, vol. 5, no. 1, pp. 55–69, Jan. 2013, publisher: American Meteorological Society Section: Weather, Climate, and Society. [Online]. Available: https://journals.ametsoc.org/view/journals/wcas/5/1/wcas-d-12-00022_1.xml 1

[20] D. Rico, C. Detweiler, and F. Munoz-Arriola, "Power-over-Tether UAS Leveraged for Nearly-Indefinite Meteorological Data Acquisition," *Annual International Meeting (AIM) of the American Society of Agricultural and Biological Engineers (ASABE)*, 2020. 1, 1.1, 6.4

[21] D. Rico, F. Munoz-Arriola, and C. Detweiler, "Trajectory Selection for Power-over-Tether Atmospheric Sensing UAS," *IEEE Intelligent RObots and Systems (IROS)*, Mar. 2021. 1.1, 6.4

[22] L. Zikou, C. Papachristos, and A. Tzes, "The Power-over-Tether system for powering small UAVs: Tethering-line tension control synthesis," in *2015 23rd Mediterranean Conference on Control and Automation (MED)*, June 2015, pp. 681–687. 2.1

[23] C. Papachristos and A. Tzes, "The power-tethered UAV-UGV team: A collaborative strategy for navigation in partially-mapped environments," in *22nd Mediterranean Conference on Control and Automation*, June 2014, pp. 1153–1158. 2.1

[24] S. Kiribayashi, K. Yakushigawa, and K. Nagatani, "Design and Development of Tether-Powered Multirotor Micro Unmanned Aerial Vehicle System for Remote-Controlled Construction Machine," in *Field and Service Robotics*, ser. Springer Proceedings in Advanced Robotics, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 637–648. 2.1

[25] F. Muttin, "Umbilical deployment modeling for tethered UAV detecting oil pollution from ship," *Applied Ocean Research*, vol. 33, no. 4, pp. 332–343, Oct. 2011. 2.1

[26] B. W. Gu, S. Y. Choi, Y. S. Choi, G. Cai, L. Seneviratne, and C. T. Rim, "Novel Roaming and Stationary Tethered Aerial Robots for Continuous Mobile Missions in Nuclear Power Plants," *Nuclear Engineering and Technology*, vol. 48, no. 4, pp. 982–996, Aug. 2016. 2.1

[27] "Hoverfly - Tethered Drone Technology For Infinite Flight Time." [Online]. Available: https://hoverflytech.com/ 2.1

[28] "Powerline Tethered Drone Systems." [Online]. Available: https://www.ntpdrone.com/ 2.1

[29] "Elistair | Tethered Drone Solutions." [Online]. Available: https://elistair.com/ 2.1

[30] D. Ibekwe, "A Latvian company is developing drones that can put out fires." [Online]. Available: https://www.businessinsider.com/aerones-firefighting-drones-2018-4 2.1

[31] W. Brown, "The History of Power Transmission by Radio Waves," *IEEE Transactions on Microwave Theory and Techniques*, vol. 32, no. 9, pp. 1230–1242, Sept. 1984, conference Name: IEEE Transactions on Microwave Theory and Techniques. 2.1

[32] M. C. Achtelik, J. Stumpf, D. Gurdan, and K.-M. Doth, "Design of a flexible high performance quadcopter platform breaking the MAV endurance record with laser power beaming," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2011, pp. 5166–5172, iSSN: 2153-0866. 2.1

[33] N. US Department of Commerce, "Education Corner weather balloon," publisher: NOAA's National Weather Service. [Online]. Available: https://www.weather.gov/gjt/education_corner_balloon 2.2

[34] S. Laroche and R. Sarrazin, "Impact of Radiosonde Balloon Drift on Numerical Weather Prediction and Verification," *Weather and Forecasting*, vol. 28, no. 3, pp. 772–782, June 2013, publisher: American Meteorological Society Section: Weather and Forecasting. 2.2

[35] "Flux Tower Measurements | NSF NEON | Open Data to Understand our Ecosystems." [Online]. Available: https://www.neonscience.org/data-collection/flux-tower-measurements 2.2

[36] "AmeriFlux: Measuring carbon, water and energy flux across the Americas." [Online]. Available: https://ameriflux.lbl.gov/ 2.2

[37] B. B. Balsley, R. G. Frehlich, M. L. Jensen, and Y. Meillier, "High-Resolution In Situ Profiling through the Stable Boundary Layer: Examination of the SBL Top in Terms of Minimum Shear, Maximum Stratification, and Turbulence Decrease," *Journal of the Atmospheric Sciences*, vol. 63, no. 4, pp. 1291–1307, Apr. 2006, publisher: American Meteorological Society Section: Journal of the Atmospheric Sciences. [Online]. Available: https://journals.ametsoc.org/view/journals/atsc/63/4/jas3671.1.xml 2.3

[38] B. B. Balsley, M. L. Jensen, and R. G. Frehlich, "The Use of State-of-the-Art Kites for Profiling the Lower Atmosphere," *Boundary-Layer Meteorology*, vol. 87, no. 1, pp. 1–25, Apr. 1998. [Online]. Available: https://doi.org/10.1023/A:1000812511429 2.3

[39] "Sensor System Suspended by Kite for Lower Troposphere Monitoring | AIAA SciTech Forum," archive Location: world. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-0597 2.3

[40] M. Bryson, M. Johnson-Roberson, R. J. Murphy, and D. Bongiorno, "Kite Aerial Photography for Low-Cost, Ultra-high Spatial Resolution Multi-Spectral Mapping of Intertidal Landscapes," *PLOS ONE*, vol. 8, no. 9, p. e73550, Sept. 2013, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0073550 2.3

[41] M. J. Irvin, "High-altitude Inflatable Kites and Their Role in Atmospheric Boundary Layer Research," *Bulletin of the American Meteorological Society*,

vol. -1, no. aop, pp. 1–13, Apr. 2021, publisher: American Meteorological Society Section: Bulletin of the American Meteorological Society. [Online]. Available: https://journals.ametsoc.org/view/journals/bams/aop/BAMS-D-20-0242.1/BAMS-D-20-0242.1.xml 2.3

[42] Kathleen McNamara, "OSU High Altitude Weather Kite HAWK," Mar. 2021. [Online]. Available: https://www.youtube.com/watch?v=5JjwpDknr2U 2.3

[43] M. T. Chilinski, K. M. Markowicz, and M. Kubicki, "UAS as a Support for Atmospheric Aerosols Research: Case Study," *Pure and Applied Geophysics*, vol. 175, no. 9, pp. 3325–3342, Sept. 2018. 2.3

[44] R. T. Palomaki, N. T. Rose, M. van den Bossche, T. J. Sherman, and S. F. J. De Wekker, "Wind Estimation in the Lower Atmosphere Using Multirotor Aircraft," *Journal of Atmospheric and Oceanic Technology*, vol. 34, no. 5, pp. 1183–1191, Apr. 2017. 2.3

[45] I. d. Boisblanc, N. Dodbele, L. Kussmann, R. Mukherji, D. Chestnut, S. Phelps, G. C. Lewin, and S. d. Wekker, "Designing a hexacopter for the collection of atmospheric flow data," in *2014 Systems and Information Engineering Design Symposium (SIEDS)*, Apr. 2014, pp. 147–152. 2.3

[46] A. Islam, A. L. Houston, A. Shankar, and C. Detweiler, "Design and Evaluation of Sensor Housing for Boundary Layer Profiling Using Multirotors," *Sensors*, vol. 19, no. 11, p. 2481, Jan. 2019. 2.3

[47] C. W. Higgins, M. G. Wing, J. Kelley, C. Sayde, J. Burnett, and H. A. Holmes, "A high resolution measurement of the morning ABL transition using distributed temperature sensing and an unmanned aircraft system," *Environmental Fluid Mechanics*, vol. 18, no. 3, pp. 683–693, June 2018. 2.3

[48] M. Odelga, P. Stegagno, H. H. Blthoff, and A. Ahmad, "A Setup for multi-UAV hardware-in-the-loop simulations," in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Nov. 2015, pp. 204–210. 2.4

[49] Q. Bu, F. Wan, Z. Xie, Q. Ren, J. Zhang, and S. Liu, "General simulation platform for vision based UAV testing," in *2015 IEEE International Conference on Information and Automation*, Aug. 2015, pp. 2512–2516. 2.4

[50] D. Santamara, F. Alarcn, A. Jimnez, A. Viguria, M. Bjar, and A. Ollero, "Model-Based Design, Development and Validation for UAS Critical Software," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 103–114, Jan. 2012. 2.4

[51] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, A. Pironti, and M. Virgilio, "Algorithms for 3D UAV Path Generation and Tracking," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 5275–5280. 2.4

[52] K. A. Talke, M. De Oliveira, and T. Bewley, "Catenary Tether Shape Analysis for a UAV - USV Team," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 7803–7809. 2.4

[53] T. Lee, "Geometric controls for a tethered quadrotor UAV," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec. 2015, pp. 2749–2754. 2.4

[54] A. L. Houston and J. M. Keeler, "The Impact of Sensor Response and Airspeed on the Representation of the Convective Boundary Layer and Airmass Boundaries by Small Unmanned Aircraft Systems," *Journal of Atmospheric and Oceanic Technology*, vol. 35, no. 8, pp. 1687–1699, Aug. 2018, publisher: American Meteorological Society Section: Journal

of Atmospheric and Oceanic Technology. [Online]. Available: https://journals.ametsoc.org/view/journals/atot/35/8/jtech-d-18-0019.1.xml 2.4

[55] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr., "Survey on Coverage Path Planning with Unmanned Aerial Vehicles," *Drones*, vol. 3, no. 1, p. 4, Mar. 2019, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. 2.4

[56] W. P. Coutinho, M. Battarra, and J. Fliege, "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review," *Computers & Industrial Engineering*, vol. 120, pp. 116–128, June 2018. 2.4

[57] "Mesonet by NSCO |." [Online]. Available: https://mesonet.unl.edu/ 3.1

[58] "OSH Park ˜." [Online]. Available: https://oshpark.com/ 3.3.1

[59] "Fusion PCB Manufacturing & Prototype PCB Assembly - Seeed Studio." [Online]. Available: https://www.seeedstudio.com/fusion_pcb.html 3.3.1

[60] *DJI - Flame Wheel F450 ARF Kit*. 3.3.1

[61] "Products | Pixhawk." [Online]. Available: https://pixhawk.org/products/ 3.3.1

[62] M. I. Bahari, P. Tarassodi, Y. M. Naeini, A. K. Khalilabad, and P. Shirazi, "Modeling and simulation of hill climbing MPPT algorithm for photovoltaic application," in *2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, June 2016, pp. 1041–1044. 3.3.3

[63] Daniel Rico, "Power-over-Tether_prototype_development," Feb. 2019. [Online]. Available: https://www.youtube.com/watch?v=48K1A4loiPo 4.2

[64] ——, "DWFI Presentation Video," May 2020. [Online]. Available: https://www.youtube.com/watch?v=QJCBAQZKGHI 4.2

[65] "BSE - Rogers Memorial Farm | College of Engineering | University of Nebraska–Lincoln." 4.3

[66] "Freyja Simulator," Apr. 2020. [Online]. Available: https://github.com/unl-nimbus-lab/Freyja-Simulator 5.2

[67] J.-P. Ore and C. Detweiler, "Sensing Water Properties at Precise Depths from the Air," *Journal of Field Robotics*, p. 37, 2018. 5.2

[68] A. Shankar, S. Elbaum, and C. Detweiler, "In-Air Exchange of Small Payloads Between Multirotor Aerial Systems," in *Proceedings of the 2018 International Symposium on Experimental Robotics*, 2018, vol. 11, pp. 511–523. 5.2

[69] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. 5.3

[70] S. A. H. Tabatabaei, A. Yousefi-koma, M. Ayati, and S. S. Mohtasebi, "Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, Oct. 2015, pp. 784–788. 5.5

[71] "Environmental Modelling & Software - Journal - Elsevier." [Online]. Available: https://www.journals.elsevier.com/journals.elsevier.com/environmental-modelling-and-software 6.4

Appendix A

Research Media

A.1 IEEE IROS-2021

Conference Paper Supplementary Upload:

https://www.youtube.com/watch?v=kFltVEmIw9I

A.2 System Development and Field Experiments

TAUS-Alpha Demonstration:

https://www.youtube.com/watch?v=XzIkagLx8WY

TAUS-Beta Pseudo-Tension Feedback Spooling Mechanism:

https://www.youtube.com/watch?v=At6fm3pX0C4

TAUS-Beta Development Cycle:

https://www.youtube.com/watch?v=1wQy2PTTjLo

TAUS-Beta Field Test at RMF:

https://www.youtube.com/watch?v=rNMXt_1eq-g

A.3 Simulations

TAUS-Beta Model Circular Trajectory Demonstration:

https://www.youtube.com/watch?v=tnmwONi1MH0

Early Progress:

https://www.youtube.com/watch?v=F-OwL5yMAdk

Volume via Convex Hull Algorithm:

https://www.youtube.com/watch?v=eStrHEGppDE

Highlights for US Strategic Air Command (US-STRATCOM) Demonstration:

https://www.youtube.com/watch?v=lFX3mppIj60

Archimedean Spiral Trajectory:

https://www.youtube.com/watch?v=drliezsqIIs

Flower Trajectory:

https://www.youtube.com/watch?v=YoDxZAAtJzU

Compilation of Lawn, Spiral, Star, and Flower Trajectories:

https://www.youtube.com/watch?v=T86fZ-j_5vE

Sensor Response to Flower Traversal in *Static* Temperature Field:

https://www.youtube.com/watch?v=mj7usE46lI8

Sensor Response to Flower Traversal in *Dynamic* Temperature Field:

https://www.youtube.com/watch?v=gEdgayQvpYw

Sensor Response to Star Traversal in *Dynamic* Temperature Field:

https://www.youtube.com/watch?v=g3wHKGVnXXY

Appendix B

Matlab Simulation Scripts

B.1 Freyja: Quad-rotor System Parameters

```
% script to generate useful quantities for the quadrotor
% simulator
global total_mass LK RYaw
total_mass = 0.700;

% System matrices
A = [ 0 0 0 1 0 0 0; ...
      0 0 0 0 1 0 0; ...
      0 0 0 0 0 1 0; ...
        zeros(4,7)  ];
B = [ zeros(3,4); ...
       eye(4) ];

% LQR params
Q = [ 10    0    0    0    0    0    0
      0    10    0    0    0    0    0
```

```
17          0       0       10       0       0       0       0
18          0       0       0       10       0       0       0
19          0       0       0       0       10       0       0
20          0       0       0       0       0       10       0
21          0       0       0       0       0       0       5 ];
22 R = [ 0.8       0       0       0
23          0       0.8       0       0
24          0       0       1       0
25          0       0       0       0.1 ];
26 R = R * 0.05;
27
28 % Feedback gain from LQR
29 LK = lqr( A, B, Q, R );
30 LKD = lqrd( A, B, Q, R, 0.01 );
31
32 % temporary fix
33 RYaw = [  cos(0) -sin(0)   0; ...
34             sin(0)  cos(0)   0;
35             0       0       1 ];
```

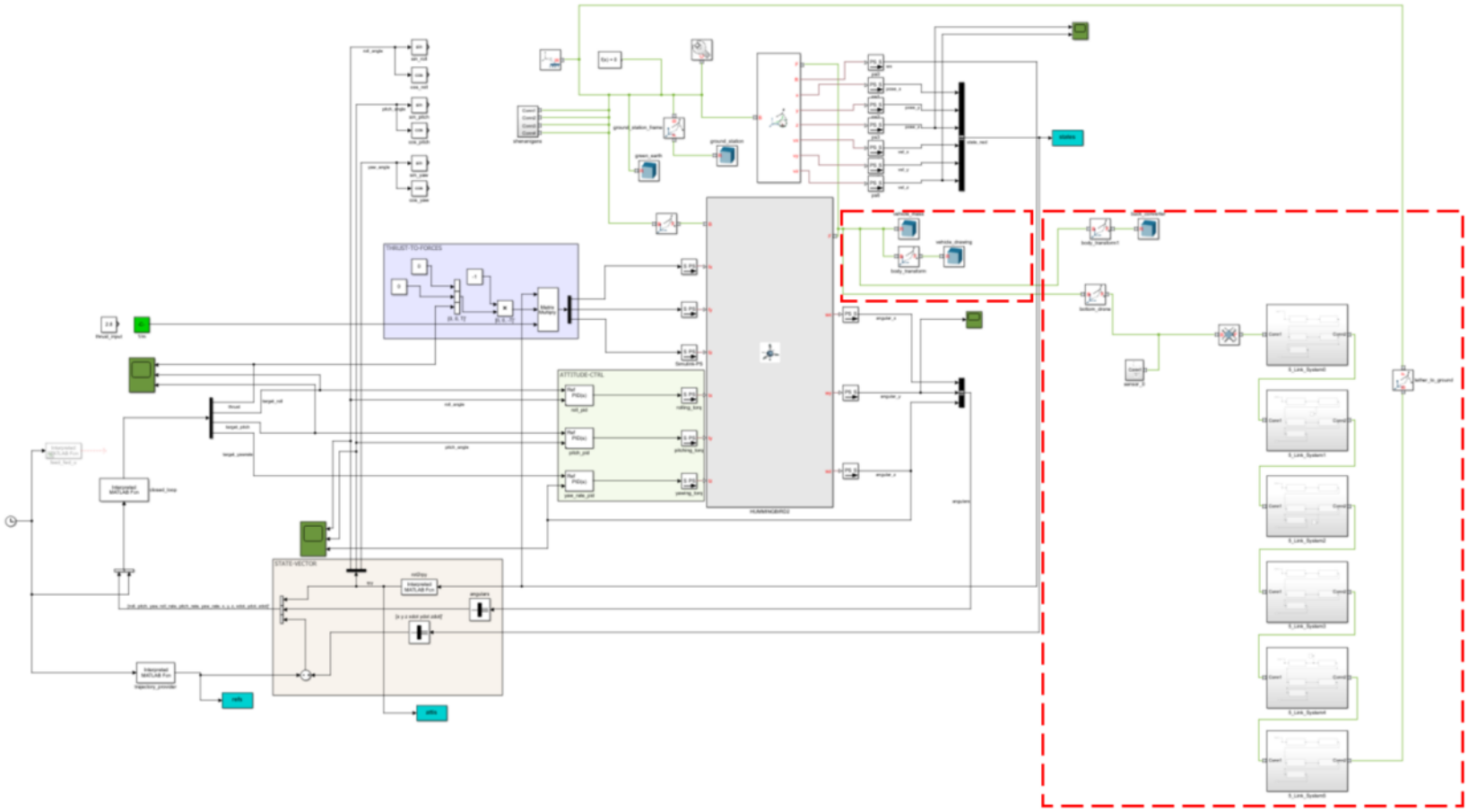


Figure B.1: The Simulink-Matlab representation of the Freyja model is presented along with the power-tether and temperature sensor modifications outlined in red.

B.2 Freyja: Power-Tether Model Parameters

```
%SOLID
link_radius = 0.01;    %[m]
link_length = 0.250;   %[m]
link_mass = 0.0005;    %[kg]

%JOINT
rx_spring_stiff = 0.000150;    %[] 0.000150;
rx_damping_coef = 0.000150;    %[]

ry_spring_stiff = 0.000150;    %[]
ry_damping_coef = 0.000150;    %[]

x_val = 0.100;
```

B.3 Freyja: Trajectory Provider

```
function traj_ref = trajectory_provider(t)

  persistent init_done w alpha_val beta_val n_dir e_dir flag
      current_time

    if isempty(init_done) %this allows for "static"
       variables to decriment
        n_dir = 0;
        e_dir = 0;
        w = 2; %was 2
        alpha_val = 1; %was 1.5
        beta_val = 1; %was 0.75
        H = 5;
        flag = 0;
        current_time = 0;

        %fprintf('\n\n***HITo***\ntime_flag_swirl_1=\t%d\nt
            =\t\t\t%.6f\nmod(t,2)=\t\t%.6f\nnum_t=\t\t\t%d\
            nalpha=\t\t\t%f\nbeta=\t\t\t%f\n***HITo***\n\n',
            time_flag_swirl_1,t,mod(t,1),num_t_flags,
            alpha_val,beta_val);
        init_done = true;
    end

```

```
19      traj_selector = 0; %0 = lawn, 1 = inscribed-lawn, 2 =
            spiral, 3 = pentagram, 4 = flower
20
21      if traj_selector == 0 %lawn
22          z = -14.5;
23          Vn = 0.5;
24          Ve = 0.5;
25          if(t>1 && t<7.5)
26              traj_ref = [-0.1923*t,-0.4615*t,z
                    ,-0.1923,-0.4615,0];
27          elseif(t>7.5 && t<8.5) %wait 1 sec
28              traj_ref = [-1.346,-3.500,z,0,0,0];
29          elseif(t>8.5 && t<13.88)
30              traj_ref = [Vn*t-5.596,-3.500,z,Vn,0,0];
31          elseif(t>13.88 && t<14.88) %wait 1 sec
32              traj_ref = [1.346,-3.500,z,0,0,0];
33          elseif(t>14.88 && t<17.68)
34              traj_ref = [1.346,Ve*t-10.94,z,0,Ve,0];
35          elseif(t>17.68 && t<18.68) %wait 1 sec
36              traj_ref = [1.346,-2.100,z,0,0,0];
37          elseif(t>18.68 && t<27.586)
38              traj_ref = [-Vn*t+10.686,-2.100,z,-Vn,0,0];
39          elseif(t>27.586 && t<28.586) %wait 1 sec
40              traj_ref = [-3.107,-2.1,z,0,0,0];
41          elseif(t>28.586 && t<31.393)
```

```
42              traj_ref = [-3.107,Ve*t-16.393,z,0,Ve,0];
43          elseif(t>31.393 && t<32.39) %wait 1 sec
44              traj_ref = [-3.107,-0.700,z,0,0,0];
45          elseif(t>32.39 && t<45.97)
46              traj_ref = [Vn*t-19.3,-0.700,z,Vn,0,0];
47          elseif(t>45.97 && t<46.97) %wait 1 sec
48              traj_ref = [3.684,-0.700,z,0,0,0];
49          elseif(t>46.97 && t<49.77)
50              traj_ref = [3.684,Ve*t-24.185,z,0,Ve,0];
51          elseif(t>49.77 && t<50.77) %wait 1 sec
52              traj_ref = [3.684,0.700,z,0,0,0];
53          elseif(t>50.77 && t<64.352)
54              traj_ref = [-Vn*t+29.069,0.700,z,-Vn,0,0];
55          elseif(t>64.352 && t<65.352) %wait 1 sec
56              traj_ref = [-3.107,0.700,z,0,0,0];
57          elseif(t>65.352 && t<68.152)
58              traj_ref = [-3.107,Ve*t-31.976,z,0,Ve,0];
59          elseif(t>68.152 && t<69.152)
60              traj_ref = [-3.107,2.100,z,0,0,0];
61          elseif(t>69.152 && t<78.058)
62              traj_ref = [Vn*t-37.683,2.100,z,Vn,0,0];
63          elseif(t>78.058 && t<79.058)
64              traj_ref = [1.346,2.100,z,0,0,0];
65          elseif(t>79.058 && t<81.858)
66              traj_ref = [1.346,Ve*t-37.429,z,0,Ve,0];
```

```
67          elseif(t>81.858 && t<82.858)
68              traj_ref = [1.346,3.500,z,0,0,0];
69          elseif(t>82.858 && t<88.242)
70              traj_ref = [-Ve*t+42.775,3.500,z,-Vn,0,0];
71          elseif(t>88.242 && t<89.242)
72              traj_ref = [-1.346,3.500,z,0,0,0];
73          else
74              traj_ref = [0,0,z,0,0,0];
75          end
76      elseif traj_selector == 1 %inscribed-lawn condition
77          z = -14.5;
78          Vn = 0.5;
79          Ve = 0.5;
80          if(t>1 && t<7.5) %get to back left corner...velocity
                is combination of both x and y (0.707 m/s)...fix
                this...
81              traj_ref = [-0.35355*t,-0.35355*t,z
                   ,-0.35355,-0.35355,0];
82          elseif(t>7.5 && t<8.5) %wait 1 sec
83              traj_ref = [-2.65,-2.65,z,0,0,0];
84          elseif(t>8.5 && t<19.1) %alpha (top left)
85              traj_ref = [Vn*t-6.9,-2.65,z,Vn,0,0];
86          elseif(t>19.1 && t<20.1) %wait 1 sec
87              traj_ref = [2.65,-2.65,z,0,0,0];
88          elseif(t>20.1 && t<22.75) %beta move
```

```
89              traj_ref = [2.65,Ve*t-12.7,z,0,Ve,0];
90          elseif(t>22.75 && t<23.75) %wait 1 sec
91              traj_ref = [2.65,-1.325,z,0,0,0];
92          elseif(t>23.75 && t<34.35) %alpha (bottom middle)
93              traj_ref = [-Vn*t+14.525,-1.325,z,-Vn,0,0]; %
94          elseif(t>34.35 && t<35.35) %wait 1 sec
95              traj_ref = [-2.65,-1.325,z,0,0,0];
96          elseif(t>35.35 && t<38) %beta move
97              traj_ref = [-2.65,Ve*t-19,z,0,Ve,0];
98          elseif(t>38 && t<39) %wait 1 sec
99              traj_ref = [-2.65,0,z,0,0,0];
100         elseif(t>39 && t<49.6) %alpha (top middle)
101             traj_ref = [Vn*t-22.15,0,z,Vn,0,0];
102         elseif(t>49.6 && t<50.6) %wait 1 sec
103             traj_ref = [2.65,0,z,0,0,0];
104         elseif(t>50.6 && t<53.25) %beta move
105             traj_ref = [2.65,Ve*t-25.3,z,0,Ve,0];
106         elseif(t>53.25 && t<54.25) %wait 1 sec
107             traj_ref = [2.65,1.325,z,0,0,0];
108         elseif(t>54.25 && t<64.85) %alpha (bottom right)
109             traj_ref = [-Vn*t+29.775,1.325,z,-Vn,0,0];
110         elseif(t>64.85 && t<65.85) %wait 1 sec
111             traj_ref = [-2.65,1.325,z,0,0,0];
112         elseif(t>65.85 && t<68.5) %beta move %
113             traj_ref = [-2.65,Ve*t-31.6,z,0,Ve,0];
```

```
114         elseif(t>68.5 && t<69.5) %wait 1 sec
115             traj_ref = [-2.65,2.65,z,0,0,0];
116         elseif(t>69.5 && t<80.1) %alpha (bottom right)
117             traj_ref = [Vn*t-37.4,2.65,z,Vn,0,0];
118         elseif(t>80.1 && t<81.1) %wait 1 sec
119             traj_ref = [2.65,2.65,z,0,0,0];
120         else
121             traj_ref = [0,0,z,0,0,0];
122         end
123
124     elseif traj_selector == 2 %spiral conditions
125         z = -14.5;
126         th = sqrt(2)*sqrt((0.5*t)/0.1193662073);
127         th_max = 0.1338688086*(t-118.3636)+31.4897; %where
              the other leaves off exactly...radian wise...
128         r = 0.1193662073*th;
129         r_max = 0.1193662073*(sqrt(2)*sqrt((0.5*118.3636)
              /0.1193662073));
130         if(t>1 && t<118.3636)
131             traj_ref = [r*sin(th),r*cos(th),z,...
132         ((0.172747*sin(2.89441*sqrt(t)))/((sqrt(t))))+(0.5*
              cos(2.89441*sqrt(t))),...
133         ((0.172747*cos(2.89441*sqrt(t)))/((sqrt(t))))-(0.5*
              sin(2.89441*sqrt(t))),...
134         0];
```

```
135         elseif(t>118.3636 && t <165.298)
136             traj_ref = [r_max*sin(th_max),r_max*cos(th_max),
                   z,0.503186*cos(0.133869*(t-118.3636)+31.4897)
                   ,-0.503186*sin(0.133869*(t-118.3636)+31.4897)
                   ,0];
137         else
138             traj_ref = [0,0,z,0,0,0];
139         end
140
141     elseif traj_selector == 3 %star conditions
142         z = -14.5;
143         half_sigma = 0.62831;
144         omega = 0.314159;
145         psi = 1.2566;
146         mu = 0.62833;
147         delta = 0.94246;
148
149         if(t>1 && t<7.5) %0,0 to bottom left...
150             traj_ref = [-0.404511*t,-0.293889*t,z,-0.5*cos(
                   half_sigma),-0.5*sin(half_sigma),0];
151         elseif(t>7.5 && t<8.5)
152             traj_ref = [-3.0338,-2.20419,z,0,0,0];
153         elseif(t>8.5 && t<22.765)
154             traj_ref = [0.5*cos(omega)*t-7.0758,0.5*sin(
                   omega)*t-3.51749,z,0.5*cos(omega),0.5*sin(
```

```
            omega),0];
155         elseif(t>22.765 && t<23.765)
156             traj_ref = [3.75,0,z,0,0,0];
157         elseif(t>23.765 && t<38.030)
158             traj_ref = [-0.5*sin(psi)*t+15.05079,0.5*cos(psi
                )*t-3.67231,z,-0.5*sin(psi),0.5*cos(psi),0];
159         elseif(t>38.030 && t<39.030)
160             traj_ref = [-3.0338,2.20419,z,0,0,0];
161         elseif(t>39.030 && t<53.295)
162             traj_ref = [0.5*sin(mu)*t-14.5046,-0.5*cos(mu)*t
                +17.992,z,0.5*sin(mu),-0.5*cos(mu),0];
163         elseif(t>53.295 && t<54.295)
164             traj_ref = [1.1588137,-3.5664619,z,0,0,0];
165         elseif(t>54.295 && t<68.56)
166             traj_ref = [1.11588137,0.5*t-30.71396,z
                ,0,0.5,0];
167         elseif(t>68.56 && t<69.56)
168             traj_ref = [1.11588137,3.5664619,z,0,0,0];
169         elseif(t>69.56 && t<83.825)
170             traj_ref = [-0.5*cos(delta)*t+21.5596,-0.5*sin(
                delta)*t+31.7037,z,-0.5*cos(delta),-0.5*sin(
                delta),0];
171         elseif(t>83.825 && t<84.825)
172             traj_ref = [-3.0338,-2.20419,z,0,0,0];
173         else
```

```
174             traj_ref = [0,0,z,0,0,0];
175         end
176
177     elseif traj_selector == 4 %flower conditions
178         z = -14.5;
179         eta = (2*3.14159)/23.56;
180
181         if(t>1 && t<23.56)
182             traj_ref = [1.875*sin(eta*t-3.14159),1.875*cos(
                    eta*t-3.14159)+1.875,z,0.500063*cos(0.2667*t
                    -3.14159),-0.500063*sin(0.2667*t-3.14159),0];
                     %RIGHT
183         elseif(t>23.56 && t<47.12)
184             traj_ref = [1.875*sin(eta*t-23.56-3.92699)
                    +1.326,1.875*cos(eta*t-23.56-3.92699)+1.326,z
                    ,0.500063*cos(0.2667*t-23.56-3.92699)
                    ,-0.500063*sin(0.2667*t-23.56-3.92699),0]; %
                    UP-RIGHT
185         elseif(t>47.12 && t<70.68)
186             traj_ref = [1.875*sin(eta*t-47.12-4.71239)
                    +1.875,1.875*cos(eta*t-47.12-4.71239),z
                    ,0.500063*cos(0.2667*t-47.12-4.71239)
                    ,-0.500063*sin(0.2667*t-47.12-4.71239),0]; %
                    UP
187         elseif(t>70.68 && t<94.24)
```

```
188             traj_ref = [1.875*sin(eta*t-70.68-5.49779)
                    +1.326,1.875*cos(eta*t-70.68-5.49779)-1.326,z
                    ,0.500063*cos(0.2667*t-70.68-5.49779)
                    ,-0.500063*sin(0.2667*t-70.68-5.49779),0]; %
                    UP-LEFT
189         elseif(t>94.24 && t<117.80)
190             traj_ref = [1.875*sin(eta*t-94.24-6.28319)
                    ,1.875*cos(eta*t-94.24-6.28319)-1.875,z
                    ,0.500063*cos(0.2667*t-94.24-6.28319)
                    ,-0.500063*sin(0.2667*t-94.24-6.28319),0]; %
                    LEFT
191         elseif(t>117.80 && t<141.36)
192             traj_ref = [1.875*sin(eta*t-117.80-7.06859)
                    -1.326,1.875*cos(eta*t-117.80-7.06859)-1.326,
                    z,0.500063*cos(0.2667*t-117.80-7.06859)
                    ,-0.500063*sin(0.2667*t-117.80-7.06859),0]; %
                    DOWN-LEFT
193         elseif(t>141.36 && t<164.92)
194             traj_ref = [1.875*sin(eta*t-141.36-7.85399)
                    -1.875,1.875*cos(eta*t-141.36-7.85399),z
                    ,0.500063*cos(0.2667*t-141.36-7.85399)
                    ,-0.500063*sin(0.2667*t-141.36-7.85399),0]; %
                    DOWN
195         elseif(t>164.92 && t<188.48)
196             traj_ref = [1.875*sin(eta*t-164.92-8.63939)
```

```
                    -1.326,1.875*cos(eta*t-164.92-8.63939)+1.326,
                    z,0.500063*cos(0.2667*t-164.92-8.63939)
                    ,-0.500063*sin(0.2667*t-164.92-8.63939),0]; %
                    DOWN-RIGHT
197             else
198                 traj_ref = [0,0,z,0,0,0];
199             end
200
201         end %end traj selector
202
203 end %end traj function
```

B.4 Post-Processing: Spatiotemporal Data

```
%~~~~~~~~~~~~~~~~~~clean up the workspace...
close all
clc

clear raw_samples prompt_1 downsample_factor prompt_2
    show_hull prompt_3 view_setting vol_all
clear x y z row;
clear e i j s t increment;
clear drone_x_data drone_y_data drone_z_data;
clear drone_x_line_plot drone_y_line_plot drone_z_line_plot;
clear so_x_data so_y_data so_z_data;
clear s1_x_data s1_y_data s1_z_data;
clear s2_x_data s2_y_data s2_z_data;
clear s3_x_data s3_y_data s3_z_data;
clear so_3_x_data so_3_y_data so_3_z_data;
clear t_data sample_location sample_location_log
    visited_locations;
clear distance_to_samp_so distance_to_samp_s1
    distance_to_samp_s2 distance_to_samp_s3;
clear min_ch_points percent_coverage percent_coverage_F30
    F30_flag percent_coverage_L30 L30_flag;
clear ch_vol_points_all k_all vol_all vol_perc_of_total;
clear samp_x_line_plot samp_y_line_plot samp_z_line_plot;

```

```
21 clear sim_duration energy_usage num_sensors data_throughput;
22 clear dur_str vol_str perc_str F30_str L30_str ergy_str
     data_str leg;
23 %~~~~~~~~~~~~~~~~~~clean up the workspace...
24
25 %how many samples are available (via tout number of elements
     ) and present to user
26 raw_samples = numel(tout); %numel is a 'number of elements'
     function
27 fprintf('[POST TRAJECTORY ANALYZER]\n\n');
28 fprintf('(Step 1 of 3)\n\nIs this simulation tethered with
     an array of 4 sensors? (1 = yes, 0 = no):'); %issue with
     1 (no downsampling...)
29 prompt_0 = ' ';
30 tether_y_n = input(prompt_0);
31
32 fprintf('\nWhat trajectory was simulated? (0 = lawn, 1 =
     lawn (inscribed), 2 = spiral, 3 = star, 4 = flower):'); %
     issue with 1 (no downsampling...)
33 prompt_00 = ' ';
34 trajectory = input(prompt_00);
35
36 if tether_y_n == 1
37     num_sensors = 4;
38     avg_power = 360; %24V*15A...
```

```
39 elseif tether_y_n == 0
40     num_sensors = 1;
41     avg_power = 120; %12v*10A...
42 end
43
44 fprintf('\nThe total number of raw samples logged from
     Simulink simulation: %d\n\n',raw_samples);
45 fprintf('Please enter a Down-Sample-Divising-Factor (number
     between 1 and %d):',raw_samples); %issue with 1 (no
     downsampling...) 150...250 typical?
46 prompt_1 = ' ';
47 downsample_factor = input(prompt_1);
48
49 min_ch_points_mark = 0;
50 if downsample_factor == 1
51     min_ch_points_mark = 100; %arbitrary...empirical?
52 elseif downsample_factor == 2
53     min_ch_points_mark = 50; %arbitrary...empirical?
54 elseif downsample_factor > 2
55     min_ch_points_mark = 25; %arbitrary...empirical?
56 end
57
58 fprintf('\nWould you like to see the convex hull in the
     figure? (yes = 1, no = 0):');
59 prompt_2 = ' ';
```

```
60 show_hull = input(prompt_2);
61
62 fprintf('\nWould you like to see anything plotted? (yes = 1,
       no = 0):');
63 prompt_3 = ' ';
64 show_plot = input(prompt_3);
65
66 if show_plot == 1
67     clc %clear Command Window
68     fprintf('(Step 2 of 3)\n\nView Setting:\n\n1 --> view
         (-37.5,30)\t[default matlab]\n2 --> view(180,0)\t\t[
         side view]\n3 --> view(67.5,-30)\t[angled view]\n4
         --> view(90,-90)\t\t[top view]\n5 --> view(0,-180)\t\
         t[flipped z]\n\n:');
69     prompt_3 = ' ';
70     view_setting = input(prompt_3);
71     vol_all = 0;
72 else
73     view_setting = 1; %generic needed...
74     vol_all = 0;
75     clc
76 end
77
78 if show_plot == 1
79     if view_setting == 1
```

```
80          view(-37.5,30) %https://www.mathworks.com/help/
                matlab/ref/view.html
81          axis manual
82      elseif view_setting == 2
83          view(180,0)
84          axis manual
85      elseif view_setting == 3
86          view(67.5,-30)
87          axis manual
88      elseif view_setting == 4
89          view(90,-90)
90          axis equal
91      elseif view_setting == 5
92          view(0,-180)
93          axis manual
94      end
95  end
96
97  clc %clear command window
98  if show_plot == 1
99      hold on
100     grid on
101 end
102
103 if trajectory == 1
```

```
104     static_sampling_points = 735; %with a 2.65m (+&-x) by
          2.65m (+&-y) by 14m (16-1 floor null) --> 7*7*15
          because 1x1x1 locations
105     xyz_grid = zeros(static_sampling_points,1); %n choose k
          --> (n/k) = #states 0:3m = 4 ^ (#choices x,y,z = 3)
106     increment = 1;
107     row = 1;
108     for x = -3:3
109         for y = -3:3
110             for z = 1:15
111                 xyz_grid(row,1) = x/increment;
112                 xyz_grid(row,2) = y/increment;
113                 xyz_grid(row,3) = z/increment;
114                 row = row + 1;
115             end
116         end
117     end
118     %thus
119     avail_vol = 233; %since 2.65/3.75 = 0.7067, *330 =
          233...scaled avail vol... %5.3*5.3*14.5; %this is
          what it can possibly reach...%make this dynamic...?
120
121 elseif trajectory == 0 || trajectory == 2 || trajectory == 3
      || trajectory == 4 %spiral or star or flower
122     static_sampling_points = 1215; % 9*9*15 because 1x1x1
```

```
        locations (if cube...but paraboloid reachable for
        these ...therefore 330...or better yet a cylinder...
        due to slack...)
123     xyz_grid = zeros(static_sampling_points,1); %n choose k
        --> (n/k) = #states 0:3m = 4 ^ (#choices x,y,z = 3)
124     increment = 1;
125     row = 1;
126     for x = -4:4
127         for y = -4:4
128             for z = 1:15
129                 xyz_grid(row,1) = x/increment;
130                 xyz_grid(row,2) = y/increment;
131                 xyz_grid(row,3) = z/increment;
132                 row = row + 1;
133             end
134         end
135     end
136     %thus
137     avail_vol = 330; %7.5*7.5*14.5 (if cube...but paraboloid
          at max...330); %this is what it can possibly reach
        ...%make this dynamic...
138 end
139
140 if show_plot == 1
141     for i = 1:static_sampling_points
```

```
142         plot3(xyz_grid(i,1),xyz_grid(i,2),xyz_grid(i,3),'s',
                'MarkerSize',12,'Color','b') %might consider
                using size 9
143     end
144
145     if trajectory == 1
146         xlim([-4,4])
147         ylim([-4,4])
148         zlim([0,16])      %if positive and flipped, flipped
                data...presented...
149     elseif trajectory == 0 || trajectory == 2 || trajectory
           == 3 || trajectory == 4
150         xlim([-4,4])
151         ylim([-4,4])
152         zlim([0,16])
153     end
154
155     if tether_y_n == 1 && trajectory == 0
156         title('4sensor-6pass-LawnMower-Traversal (7.5m x 7.5
              m x 14.5m)')
157     elseif tether_y_n == 1 && trajectory == 1
158         title('4sensor-5pass-LawnMower-Traversal (5.3m x 5.3
              m x 14.5m)')
159     elseif tether_y_n == 1 && trajectory == 2
160         title('4sensor-5+1pass-Spiral-Traversal (7.5m x 7.5m
```

```
            x 14.5m)')
161     elseif tether_y_n == 1 && trajectory == 3
162         title('4sensor-5pass-Star-Traversal (7.5m x 7.5m x
              14.5m)')
163     elseif  tether_y_n == 1 && trajectory == 4
164         title('4sensor-8lobe-Flower-Traversal (7.5m x 7.5m x
              14.5m)')
165
166     elseif tether_y_n == 0 && trajectory == 0
167         title('1sensor-6pass-LawnMower-Traversal (7.5m x 7.5
              m x 14.5m)')
168     elseif tether_y_n == 0 && trajectory == 1
169         title('1sensor-5pass-LawnMower-Traversal (5.3m x 5.3
              m x 14.5m)')
170     elseif tether_y_n == 0 && trajectory == 2
171         title('1sensor-5+1pass-Spiral-Traversal (7.5m x 7.5m
              x 14.5m)')
172     elseif tether_y_n == 0 && trajectory == 3
173         title('1sensor-5pass-Star-Traversal (7.5m x 7.5m x
              14.5m)')
174     elseif tether_y_n == 0 && trajectory == 4
175         title('1sensor-8lobe-Flower-Traversal (7.5m x 7.5m x
              14.5m)')
176     end
177
```

```
178       xlabel('Position x [m]')
179       ylabel('Position y [m]')
180       zlabel('Position z [m]')
181
182       clc %clear Command Window
183       fprintf('(Step 3 of 3)\n\nNumber of Data Points:\t\t%d\
            nDown-Sample Factor:\t\t\t%d\nDown-Sample Data Points
            :\t%d\n\nYou have 15 seconds to start screen capture
            of Figure 1 if you wanted...',raw_samples,
            downsample_factor,floor(raw_samples/downsample_factor
            ));
184       pause(15); %might want to increase this...?
185       clc %clear Command Window
186   end
187
188   tic %start performance timer
189   %pre-allocation
190   drone_x_data = zeros(1,floor(raw_samples/downsample_factor))
        ;
191   drone_y_data = zeros(1,floor(raw_samples/downsample_factor))
        ;
192   drone_z_data = zeros(1,floor(raw_samples/downsample_factor))
        ;
193
194   so_x_data = zeros(1,floor(raw_samples/downsample_factor));
```

```
195 so_y_data = zeros(1,floor(raw_samples/downsample_factor));
196 so_z_data = zeros(1,floor(raw_samples/downsample_factor));
197
198 s1_x_data = zeros(1,floor(raw_samples/downsample_factor));
199 s1_y_data = zeros(1,floor(raw_samples/downsample_factor));
200 s1_z_data = zeros(1,floor(raw_samples/downsample_factor));
201
202 s2_x_data = zeros(1,floor(raw_samples/downsample_factor));
203 s2_y_data = zeros(1,floor(raw_samples/downsample_factor));
204 s2_z_data = zeros(1,floor(raw_samples/downsample_factor));
205
206 s3_x_data = zeros(1,floor(raw_samples/downsample_factor));
207 s3_y_data = zeros(1,floor(raw_samples/downsample_factor));
208 s3_z_data = zeros(1,floor(raw_samples/downsample_factor));
209
210 so_3_x_data = zeros(length(so_x_data*num_sensors),1); %4 is
      for num sensors...
211 so_3_y_data = zeros(length(so_y_data*num_sensors),1);
212 so_3_z_data = zeros(length(so_z_data*num_sensors),1);
213
214 t_data = zeros(1,floor(raw_samples/downsample_factor));
215
216 sample_location_log = zeros(static_sampling_points,1);
217
218 for j = 1:floor(raw_samples/downsample_factor)
```

```
219     drone_x_data(j) = states.pose_x.Data(j*downsample_factor
            );
220     drone_y_data(j) = states.pose_y.Data(j*downsample_factor
            );
221     drone_z_data(j) = states.pose_z.Data(j*downsample_factor
            );
222     t_data(j) = states.pose_z.Time(j*downsample_factor); %
            could be x or y as well... time between downsampled
            points...
223
224     so_x_data(j) = vol_sensor_0_data.so_x.Data(j*
            downsample_factor);
225     so_y_data(j) = vol_sensor_0_data.so_y.Data(j*
            downsample_factor);
226     so_z_data(j) = vol_sensor_0_data.so_z.Data(j*
            downsample_factor);
227
228     if tether_y_n == 1
229         s1_x_data(j) = vol_sensor_1_data.s1_x.Data(j*
                downsample_factor);
230         s1_y_data(j) = vol_sensor_1_data.s1_y.Data(j*
                downsample_factor);
231         s1_z_data(j) = vol_sensor_1_data.s1_z.Data(j*
                downsample_factor);
232
```

```
233             s2_x_data(j) = vol_sensor_2_data.s2_x.Data(j*
                    downsample_factor);
234             s2_y_data(j) = vol_sensor_2_data.s2_y.Data(j*
                    downsample_factor);
235             s2_z_data(j) = vol_sensor_2_data.s2_z.Data(j*
                    downsample_factor);
236
237             s3_x_data(j) = vol_sensor_3_data.s3_x.Data(j*
                    downsample_factor);
238             s3_y_data(j) = vol_sensor_3_data.s3_y.Data(j*
                    downsample_factor);
239             s3_z_data(j) = vol_sensor_3_data.s3_z.Data(j*
                    downsample_factor);
240         end
241     end
242
243         %this next section is for stacking the sensors x,y,z for
                volume
244         %correctly...
245         t = 1;
246         for s = 1:length(so_x_data)%could be y or z...
247             so_3_x_data(t,1) = so_x_data(1,s);
248             so_3_y_data(t,1) = so_y_data(1,s);
249             so_3_z_data(t,1) = so_z_data(1,s);
250             t = t + 1;
```

```
251
252         so_3_x_data(t,1) = s1_x_data(1,s);
253         so_3_y_data(t,1) = s1_y_data(1,s);
254         so_3_z_data(t,1) = s1_z_data(1,s);
255         t = t + 1;
256
257         so_3_x_data(t,1) = s2_x_data(1,s);
258         so_3_y_data(t,1) = s2_y_data(1,s);
259         so_3_z_data(t,1) = s2_z_data(1,s);
260         t = t + 1;
261
262         so_3_x_data(t,1) = s3_x_data(1,s);
263         so_3_y_data(t,1) = s3_y_data(1,s);
264         so_3_z_data(t,1) = s3_z_data(1,s);
265         t = t + 1;
266     end
267
268     %more declarations
269     min_ch_points = 0;
270     percent_coverage_F30 = 0;
271     F30_flag = 0;
272     percent_coverage_L30 = 0;
273     L30_flag = 0;
274
275     radius_xy = 0;
```

```
276     speed = 0;
277
278     radius_flag1 = 0;
279     radius_flag2 = 0;
280     radius_flag3 = 0;
281     radius_flag4 = 0;
282     radius_flag5 = 0;
283
284     for i = 2:floor(raw_samples/downsample_factor) %2
          because the first few data points are erroneous...
285
286       if trajectory == 0 || trajectory == 1 || trajectory
            == 2 || trajectory == 3 || trajectory == 4
287         clc %clear Command Window
288         radius_xy = sqrt((drone_x_data(i))^2+(
              drone_y_data(i))^2);
289         %omega = (2*pi*5)/(118.3636);
290         omega = sqrt(2)*sqrt((0.5*t_data(i))/0.119);
291         other_radius = 0.119*omega;
292         %speed = (omega/t_data(i))*other_radius;
293         speed = sqrt((states.vel_x.Data(i*
              downsample_factor)^2)+(states.vel_y.Data(i*
              downsample_factor)^2));
294
295         if radius_xy >= 0.75 && radius_flag1 == 0
```

```
296                 radius_flag1 = 1;
297                 %line([0,drone_x_data(i)],[0,drone_y_data(i)
                        ],[14.5,14.5],'Color','black','LineWidth
                        ',1);
298             elseif radius_xy >= 1.50 && radius_flag2 == 0
299                 radius_flag2 = 1;
300                 %line([0,drone_x_data(i)],[0,drone_y_data(i)
                        ],[14.5,14.5],'Color','black','LineWidth
                        ',1);
301             elseif radius_xy >= 2.25 && radius_flag3 == 0
302                 radius_flag3 = 1;
303                 %line([0,drone_x_data(i)],[0,drone_y_data(i)
                        ],[14.5,14.5],'Color','black','LineWidth
                        ',1);
304             elseif radius_xy >= 3.00 && radius_flag4 == 0
305                 radius_flag4 = 1;
306                 %line([0,drone_x_data(i)],[0,drone_y_data(i)
                        ],[14.5,14.5],'Color','black','LineWidth
                        ',1);
307             elseif radius_xy >= 3.75 && radius_flag5 == 0
308                 radius_flag5 = 1;
309                 %line([0,drone_x_data(i)],[0,drone_y_data(i)
                        ],[14.5,14.5],'Color','black','LineWidth
                        ',1);
310             end
```

```
311         end
312         fprintf('[SCRIPT RUNNING]:\n\nRemaining Data Points
               :\t%d of %d\t\t\t[ ]\nTime(data-point-delta):\t
               %.4f\t\t\t\t[s]\nx-position(uav):\t\t%.4f\t\t\t\t
               [m]\ny-position(uav):\t\t%.4f\t\t\t\t[m]\nz-
               position(uav):\t\t%.4f\t\t\t\t[m]\nRadius_xy(uav)
               :\t\t\t%.4f\t\t\t\t[m]\nSpeed:\t\t\t\t\t%.4f\t\t\
               t\t[m/s]\n',floor(raw_samples/downsample_factor)-
               i+2,floor(raw_samples/downsample_factor),abs(
               t_data(i-1)-t_data(i)),drone_x_data(i),
               drone_y_data(i),drone_z_data(i)*-1,radius_xy,
               speed)        %-1 flips z
313         drone_x_line_plot = [drone_x_data(i-1),drone_x_data(
               i)];
314         drone_y_line_plot = [drone_y_data(i-1),drone_y_data(
               i)];
315         drone_z_line_plot = [drone_z_data(i-1),drone_z_data(
               i)];
316
317         if min_ch_points >= min_ch_points_mark %convex hull
               needs a minimum number of samples...50s good...?
               empirical...can actually solve this by scaling
               data...
318             %https://www.mathworks.com/matlabcentral/answers
                   /102258-why-do-i-receive-a-qhull-precision-
```

```
    error-initial-facet-2-is-coplanar-with-the-
    interior-point
319
320 ch_vol_points_all = [so_3_x_data(1:i*4),
    so_3_y_data(1:i*4),so_3_z_data(1:i*4)]; %*4
    because num sensors is ALREADY used for this
    so_3 data set...
321 %this make sure the convex hull is correct for 1
     sensor vs 4...
322 u = length(ch_vol_points_all(:,3)); %all of the
    z elements
323 for c = 1:u %check each z element
324     if ch_vol_points_all(c,3) == 0 %checking if
        any z dimensional position is 0
325         %if it is 0, replace it with the
            starting z position
326         %(so the convex hull does not recognize
            0 values)
327         ch_vol_points_all(c,3) = so_3_z_data
            (1,1); %this element because it it
            always right (starting point is non-
            zero)
328     end
329 end
330
```

```
331             [k_all,vol_all] = convhulln(ch_vol_points_all);
332
333 %             ch_vol_points_so = [transpose(so_x_data(1:i)),
    transpose(so_y_data(1:i)),transpose(so_z_data(1:i))];
334 %             [k_so,vol_so] = convhulln(ch_vol_points_so);
335
336 %             ch_vol_points_s1 = [transpose(s1_x_data(1:i)),
    transpose(s1_y_data(1:i)),transpose(s1_z_data(1:i))];
337 %             [k_s1,vol_s1] = convhulln(ch_vol_points_s1);
338
339 %             ch_vol_points_s2 = [transpose(s2_x_data(1:i)),
    transpose(s2_y_data(1:i)),transpose(s2_z_data(1:i))];
340 %             [k_s2,vol_s2] = convhulln(ch_vol_points_s2);
341
342 %             ch_vol_points_s3 = [transpose(s3_x_data(1:i)),
    transpose(s3_y_data(1:i)),transpose(s3_z_data(1:i))];
343 %             [k_s3,vol_s3] = convhulln(ch_vol_points_s3);
344
345             min_ch_points = min_ch_points + 1;
346         else
347             min_ch_points = min_ch_points + 1;
348         end
349
350         vol_perc_of_total = (vol_all/(avail_vol))*100;
351         fprintf("Volume Sensed:\t\t\t%.3f(%.3f%%)\t[m^3]\n",
```

```
        vol_all,vol_perc_of_total);
352
353     if show_plot == 1
354         hold on
355         line(drone_x_line_plot,drone_y_line_plot,
              drone_z_line_plot*-1,'Color','red','LineWidth
              ',2)   %-1 flips z
356         plot3(so_x_data(i),so_y_data(i),so_z_data(i)*-1,
              'o','Markersize',2,'Color','blue');
357         if tether_y_n == 1
358             plot3(s1_x_data(i),s1_y_data(i),s1_z_data(i)
                  *-1,'o','Markersize',2,'Color','green');
359             plot3(s2_x_data(i),s2_y_data(i),s2_z_data(i)
                  *-1,'o','Markersize',2,'Color','black');
360             plot3(s3_x_data(i),s3_y_data(i),s3_z_data(i)
                  *-1,'o','Markersize',2,'Color','magenta')
                  ;
361         end
362         %distance to sample location
363     end
364     for sample_location = 1:static_sampling_points %
          sample locations
365         distance_to_samp_so = sqrt((so_x_data(i)-
              xyz_grid(sample_location,1))^2+(so_y_data(i)-
              xyz_grid(sample_location,2))^2+((so_z_data(i)
```

```
        *-1)-xyz_grid(sample_location,3))^2);
366     if tether_y_n == 1
367         distance_to_samp_s1 = sqrt((s1_x_data(i)-
                xyz_grid(sample_location,1))^2+(s1_y_data
                (i)-xyz_grid(sample_location,2))^2+((
                s1_z_data(i)*-1)-xyz_grid(sample_location
                ,3))^2);
368         distance_to_samp_s2 = sqrt((s2_x_data(i)-
                xyz_grid(sample_location,1))^2+(s2_y_data
                (i)-xyz_grid(sample_location,2))^2+((
                s2_z_data(i)*-1)-xyz_grid(sample_location
                ,3))^2);
369         distance_to_samp_s3 = sqrt((s3_x_data(i)-
                xyz_grid(sample_location,1))^2+(s3_y_data
                (i)-xyz_grid(sample_location,2))^2+((
                s3_z_data(i)*-1)-xyz_grid(sample_location
                ,3))^2);
370     end
371     if distance_to_samp_so <= 0.5 %arbitrary
            successful sample distance
372         sample_location_log(sample_location,1) = 1;
373         if show_plot == 1
374             plot3(xyz_grid(sample_location,1),
                    xyz_grid(sample_location,2),xyz_grid(
                    sample_location,3),'x','MarkerSize'
```

```
                ,9,'Color','r')
375             end
376             samp_x_line_plot = [so_x_data(i),xyz_grid(
                    sample_location,1)];
377             samp_y_line_plot = [so_y_data(i),xyz_grid(
                    sample_location,2)];
378             samp_z_line_plot = [(so_z_data(i)*-1),
                    xyz_grid(sample_location,3)];
379
380             if show_plot == 1
381                 line(samp_x_line_plot,samp_y_line_plot,
                        samp_z_line_plot,'Color','blue','
                        LineWidth',1)
382             end
383         end
384         if tether_y_n == 1
385             if distance_to_samp_s1 <= 0.5 %arbitrary
                    successful sample distance
386                 sample_location_log(sample_location,1) =
                        1;
387                 if show_plot == 1
388                     plot3(xyz_grid(sample_location,1),
                            xyz_grid(sample_location,2),
                            xyz_grid(sample_location,3),'x','
                            MarkerSize',9,'Color','r')
```

```
389                 end
390                 samp_x_line_plot = [s1_x_data(i),
                        xyz_grid(sample_location,1)];
391                 samp_y_line_plot = [s1_y_data(i),
                        xyz_grid(sample_location,2)];
392                 samp_z_line_plot = [(s1_z_data(i)*-1),
                        xyz_grid(sample_location,3)];
393
394                 if show_plot == 1
395                     line(samp_x_line_plot,
                            samp_y_line_plot,samp_z_line_plot
                            ,'Color','blue','LineWidth',1)
396                 end
397             end
398
399             if distance_to_samp_s2 <= 0.5 %arbitrary
                    successful sample distance
400                 sample_location_log(sample_location,1) =
                        1;
401                 if show_plot == 1
402                     plot3(xyz_grid(sample_location,1),
                            xyz_grid(sample_location,2),
                            xyz_grid(sample_location,3),'x','
                            MarkerSize',9,'Color','r')
403                 end
```

```
404                 samp_x_line_plot = [s2_x_data(i),
                        xyz_grid(sample_location,1)];
405                 samp_y_line_plot = [s2_y_data(i),
                        xyz_grid(sample_location,2)];
406                 samp_z_line_plot = [(s2_z_data(i)*-1),
                        xyz_grid(sample_location,3)];
407
408                 if show_plot == 1
409                     line(samp_x_line_plot,
                            samp_y_line_plot,samp_z_line_plot
                            ,'Color','blue','LineWidth',1)
410                 end
411             end
412             if distance_to_samp_s3 <= 0.5 %arbitrary
                    successful sample distance
413                 sample_location_log(sample_location,1) =
                        1;
414                 if show_plot == 1
415                     plot3(xyz_grid(sample_location,1),
                            xyz_grid(sample_location,2),
                            xyz_grid(sample_location,3),'x','
                            MarkerSize',9,'Color','r')
416                 end
417                 samp_x_line_plot = [s3_x_data(i),
                        xyz_grid(sample_location,1)];
```

```
418                     samp_y_line_plot = [s3_y_data(i),
                            xyz_grid(sample_location,2)];
419                     samp_z_line_plot = [(s3_z_data(i)*-1),
                            xyz_grid(sample_location,3)];
420
421                     if show_plot == 1
422                         line(samp_x_line_plot,
                                samp_y_line_plot,samp_z_line_plot
                                ,'Color','blue','LineWidth',1)
423                     end
424                 end
425             end
426         end
427
428         visited_locations = 0;
429         number_locations = static_sampling_points; %removed
              the bottom ground points...null condition
430         for e = 1:static_sampling_points
431             if sample_location_log(e,1) == 1
432                 visited_locations = visited_locations + 1;
433             end
434         end
435         percent_coverage = visited_locations/330; %
              number_locations; 330 is viable for paraboloid...
              but what about inscribed trajectory?
```

```
436         fprintf('Percent Coverage:\t\t%.3f\t\t\t\t[%%]\n',
               percent_coverage*100);
437
438         %this can fail at great downsampling...only 0.000
               will output...
439         %works great at low downsampling (<=35)
440         if floor(t_data(i)) == 30 && F30_flag == 0 %FIRST 30
               seconds reached
441             percent_coverage_F30 = percent_coverage;
442             F30_flag = 1; %prevent more than 1 entry
443            %test_F30 = 1;
444         end
445
446         %this can fail at great downsampling...only 0.000
               will output...
447         %works great at low downsampling (<=35)
448         if floor(t_data(i)) == floor((tout(length(tout))-30)
               ) && L30_flag == 0 %LAST 30 seconds reached
449             percent_coverage_L30 = percent_coverage;
450             L30_flag = 1; %prevent more than 1 entry
451            %test_L30 = 1;
452         end
453
454         %trisurf fails at downsampling <5...too similar data
455         %points...
```

```
456         if min_ch_points > min_ch_points_mark && show_hull
               == 1
457             trisurf(k_all,ch_vol_points_all(:,1),
                   ch_vol_points_all(:,2),(ch_vol_points_all
                   (:,3)*-1),'FaceColor','c');
458
459             %trisurf(k_so,ch_vol_points_so(:,1),
                   ch_vol_points_so(:,2),ch_vol_points_so(:,3)
                   *-1,'FaceColor','cyan')
460
461             %trisurf(k_s1,ch_vol_points_s1(:,1),
                   ch_vol_points_s1(:,2),ch_vol_points_s1(:,3)
                   *-1,'FaceColor','cyan')
462
463             %trisurf(k_s2,ch_vol_points_s2(:,1),
                   ch_vol_points_s2(:,2),ch_vol_points_s2(:,3)
                   *-1,'FaceColor','cyan')
464
465             %trisurf(k_s3,ch_vol_points_s3(:,1),
                   ch_vol_points_s3(:,2),ch_vol_points_s3(:,3)
                   *-1,'FaceColor','cyan')
466         end
467
468         if show_plot == 1
469             drawnow %this updates Figure 1 per iteration...
```

```
            like frame of video...
470         end
471
472         sim_duration = tout(length(tout)); %(t_data(i) is
               the increments...
473         energy_usage = (avg_power*t_data(i))/1000; %200
               Watts (average power pull from battery bank '~
               assumption')
474         data_throughput = (0.42*t_data(i)*num_sensors); %
               0.42 sampling frequency...lol...more like 1Hz...
475
476         %~~~~~~~~~~~~~~~~~~~~~~~~LEGEND
477         dur_str = {strcat('Duration [s] =',32, num2str(
               t_data(i),'%3.2f'))};                    %32 is
                ASCII for white space...
478         vol_str = {strcat('Volume [m^3] =', 32, num2str(
               vol_all,'%3.2f'), 32,'(', num2str(
               vol_perc_of_total,'%3.2f'), '%)')};
479         samp_pnt_str = {strcat('Samples [#] =', 32, num2str(
               visited_locations,'%3.2f'))};
480         perc_str = {strcat('Coverage [%] =', 32, num2str(
               percent_coverage*100,'%3.2f'))};
481         F30_str = {strcat('Cov-F30 [%] =', 32, num2str(
               percent_coverage_F30*100,'%3.2f'))};
482         L30_str = {strcat('Cov-L30 [%] =', 32, num2str(
```

```
    percent_coverage_L30*100,'%3.2f'))};
483 ergy_str = {strcat('Energy [kJ] =', 32, num2str(
    energy_usage,'%3.2f'))};                %32 is
    ASCII for white space...
484 data_str = {strcat('Data [pkts] =', 32, num2str(
    data_throughput,'%3.2f'))};
485 radius_str = {strcat('UAV Radius [m] =', 32, num2str
    (radius_xy,'%3.2f'))};
486 speed_str = {strcat('UAV Speed [m/s] =', 32, num2str
    (speed,'%3.2f'))};
487
488 if show_plot == 1
489     leg = legend(dur_str{1},vol_str{1},samp_pnt_str
        {1},perc_str{1},F30_str{1},L30_str{1},
        ergy_str{1},data_str{1},radius_str{1},
        speed_str{1},'AutoUpdate','off');
490     leg.Title.String = 'Output Data';
491     set(leg,'position',[0.2000 0.795 0.080 0.010]) %
        x,y,w,h
492    %set(leg,'position',[0.0072 0.868 0.075 0.010])
        %x,y,w,h %-0.01 because symbols dont match...
        hiding problem...
493 end
494 %~~~~~~~~~~~~~~~~~~~~~~~~LEGEND
495
```

```
496     end %major end
497
498 clc %clear Command Window
499
500 fprintf(['[OUTPUT]:\n\nNumber of Data Points:\t\t\t%d\t\t\t[
      samples]',...
501     '\nDown-Sample-Divising-Factor:\t%d\t\t\t\t[ ]',...
502     '\nDown-Sample Data Points:\t\t%d\t\t\t\t[samples]',...
503     '\nAvailable Volume:\t\t\t\t%d\t\t\t\t[m^3]',...
504     '\n\nSimulation Duration:\t\t\t%.3f\t\t\t[s]',...
505     '\nVolume Sensed:\t\t\t\t%.3f(%.3f%%)\t[m^3]',...
506     '\nSuccessfully Sampled Locations:\t%d of %d\t\t[samples
          ]',...
507     '\nPercent Coverage:\t\t\t\t%.3f\t\t\t[%%]',...
508     '\nPercent Coverage (F30):\t\t\t%.3f\t\t\t[%%]',...
509     '\nPercent Coverage (L30):\t\t\t%.3f\t\t\t[%%]',...
510     '\nEnergy Usage:\t\t\t\t%.3f\t\t\t[kJ]',...
511     '\nData:\t\t\t\t\t\t%.3f\t\t\t[pkts]\n\n'],...
512     raw_samples,downsample_factor,floor(raw_samples/
          downsample_factor),floor(avail_vol),sim_duration,
          vol_all,vol_perc_of_total,visited_locations,floor(
          avail_vol),percent_coverage*100,percent_coverage_F30
          *100,(percent_coverage*100)-(percent_coverage_L30
          *100),energy_usage,data_throughput);
513
```

```
514 toc %end performance timer (has own print statement)
515 fprintf('\n');
```

B.5 Post-Processing: Update Temperature Field

```
1
2
3 function new_temp_field = update_temp_field(temp_field,val,
    step,dimension) %dimension --> 0 = vertical [down->up], 1
    = horizontal [right->left]
4
5     for x = 1:size(temp_field,1)
6         for y = 1:size(temp_field,2)
7             for z = 1:size(temp_field,3)
8                 if dimension == 0
9                     if x <= step
10                        new_temp_field(x,y,z) = temp_field(x
                              ,y,z)-(val); %cold front (
                              vertical)
11                    end
12                    if x > step
13                        new_temp_field(x,y,z) = temp_field(x
                              ,y,z);
14                    end
15                end
16                if dimension == 1
17                    if y <= step
18                        new_temp_field(x,y,z) = temp_field(x
                              ,y,z)-(val); %cold front (
```

```
                horizontal)
19          end
20          if y > step
21              new_temp_field(x,y,z) = temp_field(x
                    ,y,z);
22          end
23      end
24      if dimension == 2
25          if x <= step
26              new_temp_field(34-x,y,z) =
                    temp_field(34-x,y,z)-(val); %cold
                     front (vertical)
27          end
28          if x > step
29              new_temp_field(34-x,y,z) =
                    temp_field(34-x,y,z);
30          end
31      end
32      if dimension == 3
33          if y <= step
34              new_temp_field(x,34-y,z) =
                    temp_field(x,34-y,z)-(val); %cold
                     front (horizontal)
35          end
36          if y > step
```

```
37                                    new_temp_field(x,34-y,z) =
                                          temp_field(x,34-y,z);
38                                end
39                            end
40                        end
41                    end
42                end
43
44 end
```

B.6 Post-Processing: Reset Temperature Field

```
1
2  function reset_temp_field = reset_temp_field(temp_field,
       select,dimension) %dimension --> 0 = vertical [down->up],
        1 = horizontal [right->left]
3
4      if select == 1
5
6          reset_temp = 3.50; %arbitrary
7
8          for x = 1:size(temp_field,1)
9              for y = 1:size(temp_field,2)
10                 for z = 1:size(temp_field,3)
11                     if dimension == 0 || dimension == 2
12                         if x <=4
13                             reset_temp_field(x,y,z) =
                                   reset_temp;
14                         end
15                         if x <= 8 && x > 4
16                             reset_temp_field(x,y,z) =
                                   reset_temp;
17                         end
18                         if x <= 12 && x > 8
19                             reset_temp_field(x,y,z) =
                                   reset_temp;
```

```
20                  end
21                  if x <= 16 && x > 12
22                      reset_temp_field(x,y,z) =
                            reset_temp;
23                  end
24                  if x <= 20 && x > 16
25                      reset_temp_field(x,y,z) =
                            reset_temp;
26                  end
27                  if x <= 24 && x > 20
28                      reset_temp_field(x,y,z) =
                            reset_temp;
29                  end
30                  if x <= 28 && x > 24
31                      reset_temp_field(x,y,z) =
                            reset_temp;
32                  end
33                  if x <= 32 && x > 28
34                      reset_temp_field(x,y,z) =
                            reset_temp;
35                  end
36                  if x <= 36 && x > 32
37                      reset_temp_field(x,y,z) =
                            reset_temp;
38                  end
```

```
39                  if x <= 40 && x > 36
40                      reset_temp_field(x,y,z) =
                            reset_temp;
41                  end
42              end
43              if dimension == 1 || dimension == 3
44                  if y <=4
45                      reset_temp_field(x,y,z) =
                            reset_temp;
46                  end
47                  if y <= 8 && y > 4
48                      reset_temp_field(x,y,z) =
                            reset_temp;
49                  end
50                  if y <= 12 && y > 8
51                      reset_temp_field(x,y,z) =
                            reset_temp;
52                  end
53                  if y <= 16 && y > 12
54                      reset_temp_field(x,y,z) =
                            reset_temp;
55                  end
56                  if y <= 20 && y > 16
57                      reset_temp_field(x,y,z) =
                            reset_temp;
```

```
58                          end
59                          if y <= 24 && y > 20
60                              reset_temp_field(x,y,z) =
                                    reset_temp;
61                          end
62                          if y <= 28 && y > 24
63                              reset_temp_field(x,y,z) =
                                    reset_temp;
64                          end
65                          if y <= 32 && y > 28
66                              reset_temp_field(x,y,z) =
                                    reset_temp;
67                          end
68                          if y <= 36 && y > 32
69                              reset_temp_field(x,y,z) =
                                    reset_temp;
70                          end
71                          if y <= 40 && y > 36
72                              reset_temp_field(x,y,z) =
                                    reset_temp;
73                          end
74                      end
75                  end
76              end
77          end
```

```
78      end
79
80
81
82
83      if select == 0
84
85          for x = 1:size(temp_field,1)
86              for y = 1:size(temp_field,2)
87                  for z = 1:size(temp_field,3)
88                      if dimension == 0 || dimension == 2
89                          if x <=4
90                              reset_temp_field(x,y,z) = 1.75;
91                          end
92                          if x <= 8 && x > 4
93                              reset_temp_field(x,y,z) = 2.00;
94                          end
95                          if x <= 12 && x > 8
96                              reset_temp_field(x,y,z) = 2.25;
97                          end
98                          if x <= 16 && x > 12
99                              reset_temp_field(x,y,z) = 2.50;
100                         end
101                         if x <= 20 && x > 16
102                             reset_temp_field(x,y,z) = 2.75;
```

```
103                 end
104                 if x <= 24 && x > 20
105                     reset_temp_field(x,y,z) = 3.00;
106                 end
107                 if x <= 28 && x > 24
108                     reset_temp_field(x,y,z) = 3.25;
109                 end
110                 if x <= 32 && x > 28
111                     reset_temp_field(x,y,z) = 3.50;
112                 end
113                 if x <= 36 && x > 32
114                     reset_temp_field(x,y,z) = 3.75;
115                 end
116                 if x <= 40 && x > 36
117                     reset_temp_field(x,y,z) = 4.00;
118                 end
119             end
120             if dimension == 1 || dimension == 3
121                 if y <=4
122                     reset_temp_field(x,y,z) = 1.75;
123                 end
124                 if y <= 8 && y > 4
125                     reset_temp_field(x,y,z) = 2.00;
126                 end
127                 if y <= 12 && y > 8
```

```
128                     reset_temp_field(x,y,z) = 2.25;
129                 end
130                 if y <= 16 && y > 12
131                     reset_temp_field(x,y,z) = 2.50;
132                 end
133                 if y <= 20 && y > 16
134                     reset_temp_field(x,y,z) = 2.75;
135                 end
136                 if y <= 24 && y > 20
137                     reset_temp_field(x,y,z) = 3.00;
138                 end
139                 if y <= 28 && y > 24
140                     reset_temp_field(x,y,z) = 3.25;
141                 end
142                 if y <= 32 && y > 28
143                     reset_temp_field(x,y,z) = 3.50;
144                 end
145                 if y <= 36 && y > 32
146                     reset_temp_field(x,y,z) = 3.75;
147                 end
148                 if y <= 40 && y > 36
149                     reset_temp_field(x,y,z) = 4.00;
150                 end
151             end
152         end
```

```
153                 end
154             end
155         end
```

B.7 Post-Processing: Sensor Response

```
clc
close all

static_or_dynamic = 1; %0 for static input, 1 for dynamic
dimension = 1; %0 = vertical [down->up], 1 = horizontal [
    left->right], 2 = vertical [up->down], 3 = horizontal [
    right->left]
sim_seconds = 84;
traj_raw_samples = 5*sim_seconds;%s; %flower = 942, lawn =
    540, spiral = 826, star = 424
model_tau = 1; %1,5,10 and 46...
temp_step_denom = 32; %original value 32...
step_increment = 1;
start_temp = 2; %2 regular, 3.50 for detection time tests...
speed_mult = 1; %speed stuff not working outside of 1...
%-------------------------------------------------------(
    mesh-interp)

[xyz_grid_x,xyz_grid_y,xyz_grid_z] = meshgrid
    (-4:0.25:4,-4:0.25:4,0:0.25:15);

temp_field = meshgrid(-4:0.25:4,-4:0.25:4,0:0.25:15);

temp_field = reset_temp_field(temp_field,0,dimension);
```

```
20
21 figure('Name','Static Temperature Field (Input)','
     NumberTitle','off')
22 slice(xyz_grid_x,xyz_grid_y,xyz_grid_z,temp_field,[0 4],4,0)
     ;
23 view(2);
24 title('Static - Temperature Field (Input)')
25 xlabel('Easting [m]','FontSize',12)
26 ylabel('Northing [m]','FontSize',12)
27 zlabel('Altitude [m]','FontSize',12)
28 set(gcf,'color','w');
29
30
31 [Xq,Yq,Zq] = meshgrid(-4:.25:4,-4:.25:4,0:.25:15);
32
33 Vq = interp3(xyz_grid_x,xyz_grid_y,xyz_grid_z,temp_field,Xq,
     Yq,Zq,'spline');
34
35 figure('Name','Static Temperature Field (Input)','
     NumberTitle','off')
36 slice(Xq,Yq,Zq,Vq,[0 4],4,0);
37 view(2);
38 title('Static - Temperature Field (Input)')
39 xlabel('Easting [m]','FontSize',12)
40 ylabel('Northing [m]','FontSize',12)
```

```
41 zlabel('Altitude [m]','FontSize',12)
42 xlim([-4 4])
43 ylim([-4 4])
44 shading flat
45 cbar = colorbar;
46 ylabel(cbar, 'Temperature [\circC]','FontSize',12)
47 set(gcf,'color','w');
48 %-----------------------------------------------------||
49
50
51 %-----------------------------------------------------(
     Raw Position)
52 figure('Name','positional data so-s3');
53 grid on
54 hold on
55 plot3(so_x_data,so_y_data,so_z_data*-1);
56 plot3(s1_x_data,s1_y_data,s1_z_data*-1);
57 plot3(s2_x_data,s2_y_data,s2_z_data*-1);
58 plot3(s3_x_data,s3_y_data,s3_z_data*-1);
59 title('Flower-Trajectory --> Sensor Spatial Data')
60 xlabel('Easting [m]','FontSize',12)
61 ylabel('Northing [m]','FontSize',12)
62 zlabel('Altitude [m]','FontSize',12)
63 xlim([-4 4])
64 ylim([-4 4])
```

```
65 legend('s4','s3','s2','s1')
66 set(gcf,'color','w');
67 %-------------------------------------------------------||
68
69
70 %-------------------------------------------------------(
       Sensor Response)
71
72 temp_field_LUT = flip(temp_field);
73
74 field_position_x = (-4:0.25:4);%[-2,-1,0,1,2];
75 field_position_y = (-4:0.25:4);%[-4,-3,-2,-1,0,1,2,3,4];
76
77 x_time_sec = 1:ceil(traj_raw_samples/(5*speed_mult)+1);%188
       is the seconds for the flower traj...meaning 5Hz data
       ...165 for spiral...85 for star
78 sensor_3_model_value = start_temp*ones(size(x_time_sec));
79 sensor_2_model_value = start_temp*ones(size(x_time_sec));
80 sensor_1_model_value = start_temp*ones(size(x_time_sec));
81 sensor_0_model_value = start_temp*ones(size(x_time_sec)); %
       NaN(size(x_time_sec));
82 %sensor_0_model_value(1) = 2;
83 LUT_temp_value_3 = start_temp*ones(size(x_time_sec));
84 LUT_temp_value_2 = start_temp*ones(size(x_time_sec));
85 LUT_temp_value_1 = start_temp*ones(size(x_time_sec));
```

```
86 LUT_temp_value_0 = start_temp*ones(size(x_time_sec));
87
88 min_distance_value = 100;  %arbitrary high
89 min_distance_x = 0;
90 min_distance_y = 0;
91
92 time_step = 1;
93 second_tick = 1;
94 previous_LUT_temp_value = start_temp;
95 temp_time = 1;
96 re_slice = 1;
97 step = 1;
98
99 if static_or_dynamic == 0
100     temp_field = reset_temp_field(temp_field,0,dimension);
101     temp_field_LUT = flip(temp_field);
102 else
103     temp_field = reset_temp_field(temp_field,1,dimension);
104     temp_field_LUT = flip(temp_field);
105 end
106
107 for k = 1:traj_raw_samples-2 %-2 to make even...
108     if static_or_dynamic == 1
109         if k >= re_slice
110             %disp('yes')
```

```
111             temp_field = update_temp_field(temp_field,0.100,
                    step,dimension);
112             temp_field_LUT = flip(temp_field);
113             step = step + step_increment;
114             re_slice = re_slice + s/temp_step_denom; %gives
                    ~32 temp steps... since -4 --> +4 with 0.25
                    step...
115         end
116     end
117     if k >= time_step %Simulation data is 5Hz... therefore
            seconds are every 5 data points...
118         second_tick = second_tick + 1;
119         for i = 1:33
120             for j = 1:33
121                 distance = sqrt((s3_x_data(k)-
                        field_position_x(i))^2+(s3_y_data(k)-
                        field_position_y(j))^2);
122                  if distance <= min_distance_value
123                      min_distance_value = distance;
124                      min_distance_x = i;
125                      min_distance_y = j;
126                  end
127             end
128         end
129     time_step = time_step + 5*speed_mult;
```

```
130
131     LUT_temp_value_3(second_tick) = temp_field_LUT(34-
            min_distance_y,min_distance_x); %33 flips the data...
132     if LUT_temp_value_3(second_tick) ~= LUT_temp_value_3(
            second_tick-1)
133         previous_LUT_temp_value = sensor_3_model_value(
                second_tick-1);%LUT_temp_value_3(second_tick-1);
134         temp_time = 1; %reset
135         sensor_3_model_value(second_tick) = (
                LUT_temp_value_3(second_tick)-
                previous_LUT_temp_value)*(1-exp(-(temp_time)/
                model_tau))+previous_LUT_temp_value;
136     elseif LUT_temp_value_3(second_tick) == LUT_temp_value_3
            (second_tick-1)
137         sensor_3_model_value(second_tick) = (
                LUT_temp_value_3(second_tick)-
                previous_LUT_temp_value)*(1-exp(-(temp_time)/
                model_tau))+previous_LUT_temp_value;
138         temp_time = temp_time + 1;
139     end
140     fprintf('\n\nSeconds = %d\nSensor_x = %f\nSensor_y = %f\
            nMin-Dist = %f\nField-pos-x = %d\nField-pos-y = %d\
            nLUT-temp-val = %d\n\n',second_tick-1,s3_x_data(k),
            s3_y_data(k), min_distance_value,min_distance_x,
            min_distance_y,LUT_temp_value_3(second_tick));
```

```
141     min_distance_value = 100; %arbitrary upper reset...
142     end
143 end
144
145
146
147 min_distance_value = 100;  %arbitrary high
148 min_distance_x = 0;
149 min_distance_y = 0;
150
151 time_step = 1;
152 second_tick = 1;
153 previous_LUT_temp_value = start_temp;
154 temp_time = 1;
155 re_slice = 1;
156 step = 1;
157
158 if static_or_dynamic == 0
159     temp_field = reset_temp_field(temp_field,0,dimension);
160     temp_field_LUT = flip(temp_field);
161 else
162     temp_field = reset_temp_field(temp_field,1,dimension);
163     temp_field_LUT = flip(temp_field);
164 end
165
```

```
166 for k = 1:traj_raw_samples-2 %-2 to make even...
167     if static_or_dynamic == 1
168         if k >= re_slice
169             %disp('yes')
170             temp_field = update_temp_field(temp_field,0.100,
                    step,dimension);
171             temp_field_LUT = flip(temp_field);
172             step = step + step_increment;
173             re_slice = re_slice + s/temp_step_denom; %gives
                    ~32 temp steps...since -4 --> +4 with 0.25
                    step...
174         end
175     end
176     if k >= time_step %Simulation data is 5Hz...therefore
            seconds are every 5 data points...
177         second_tick = second_tick + 1;
178         for i = 1:33
179             for j = 1:33
180                 distance = sqrt((s2_x_data(k)-
                        field_position_x(i))^2+(s2_y_data(k)-
                        field_position_y(j))^2);
181                  if distance <= min_distance_value
182                     min_distance_value = distance;
183                     min_distance_x = i;
184                     min_distance_y = j;
```

```
185                     end
186                 end
187             end
188         time_step = time_step + 5*speed_mult;
189         LUT_temp_value_2(second_tick) = temp_field_LUT(34-
                min_distance_y,min_distance_x);
190         if LUT_temp_value_2(second_tick) ~= LUT_temp_value_2(
                second_tick-1)
191             previous_LUT_temp_value = sensor_2_model_value(
                    second_tick-1);
192             temp_time = 1; %reset
193             sensor_2_model_value(second_tick) = (
                    LUT_temp_value_2(second_tick)-
                    previous_LUT_temp_value)*(1-exp(-(temp_time)/
                    model_tau))+previous_LUT_temp_value;
194         elseif LUT_temp_value_2(second_tick) == LUT_temp_value_2
                (second_tick-1)
195             sensor_2_model_value(second_tick) = (
                    LUT_temp_value_2(second_tick)-
                    previous_LUT_temp_value)*(1-exp(-(temp_time)/
                    model_tau))+previous_LUT_temp_value;
196             temp_time = temp_time + 1;
197         end
198         fprintf("\n\nSeconds = %d\nSensor_x = %f\nSensor_y = %f\
                nMin-Dist = %f\nField-pos-x = %d\nField-pos-y = %d\
```

```
        nLUT-temp-val = %d\n\n",second_tick-1,s2_x_data(k),
        s2_y_data(k), min_distance_value,min_distance_x,
        min_distance_y,LUT_temp_value_2(second_tick));
199     min_distance_value = 100; %arbitrary upper reset...
200     end
201 end
202
203
204
205 min_distance_value = 100;  %arbitrary high
206 min_distance_x = 0;
207 min_distance_y = 0;
208 re_slice = 1;
209 step = 1;
210
211 time_step = 1;
212 second_tick = 1;
213 previous_LUT_temp_value = start_temp;
214 temp_time = 1;
215
216 if static_or_dynamic == 0
217     temp_field = reset_temp_field(temp_field,0,dimension);
218     temp_field_LUT = flip(temp_field);
219 else
220     temp_field = reset_temp_field(temp_field,1,dimension);
```

```
221     temp_field_LUT = flip(temp_field);
222 end
223
224 for k = 1:traj_raw_samples-2 %-2 to make even...
225     if static_or_dynamic == 1
226         if k >= re_slice
227             %disp('yes')
228             temp_field = update_temp_field(temp_field,0.100,
                    step,dimension);
229             temp_field_LUT = flip(temp_field);
230             step = step + step_increment;
231             re_slice = re_slice + s/temp_step_denom; %gives
                    ~32 temp steps...since -4 --> +4 with 0.25
                    step...
232         end
233     end
234     if k >= time_step %Simulation data is 5Hz...therefore
            seconds are every 5 data points...
235         second_tick = second_tick + 1;
236         for i = 1:33
237             for j = 1:33
238                 distance = sqrt((s1_x_data(k)-
                        field_position_x(i))^2+(s1_y_data(k)-
                        field_position_y(j))^2);
239                  if distance <= min_distance_value
```

```
240                         min_distance_value = distance;
241                         min_distance_x = i;
242                         min_distance_y = j;
243                     end
244                 end
245             end
246         time_step = time_step + 5*speed_mult;
247         LUT_temp_value_1(second_tick) = temp_field_LUT(34-
              min_distance_y,min_distance_x);
248         if LUT_temp_value_1(second_tick) ~= LUT_temp_value_1(
              second_tick-1)
249             previous_LUT_temp_value = sensor_1_model_value(
                  second_tick-1);
250             temp_time = 1; %reset
251             sensor_1_model_value(second_tick) = (
                  LUT_temp_value_1(second_tick)-
                  previous_LUT_temp_value)*(1-exp(-(temp_time)/
                  model_tau))+previous_LUT_temp_value;
252         elseif LUT_temp_value_1(second_tick) == LUT_temp_value_1
              (second_tick-1)
253             sensor_1_model_value(second_tick) = (
                  LUT_temp_value_1(second_tick)-
                  previous_LUT_temp_value)*(1-exp(-(temp_time)/
                  model_tau))+previous_LUT_temp_value;
254             temp_time = temp_time + 1;
```

```
255     end
256     fprintf('\n\nSeconds = %d\nSensor_x = %f\nSensor_y = %f\
          nMin-Dist = %f\nField-pos-x = %d\nField-pos-y = %d\
          nLUT-temp-val = %d\n\n',second_tick-1,s1_x_data(k),
          s1_y_data(k), min_distance_value,min_distance_x,
          min_distance_y,LUT_temp_value_1(second_tick));
257     min_distance_value = 100; %arbitrary upper reset...
258     end
259 end
260
261
262
263 min_distance_value = 100;  %arbitrary high
264 min_distance_x = 0;
265 min_distance_y = 0;
266
267 time_step = 1;
268 second_tick = 1;
269 previous_LUT_temp_value = start_temp;
270 temp_time = 1;
271 re_slice = 1;
272 step = 1;
273
274 if static_or_dynamic == 0
275     temp_field = reset_temp_field(temp_field,0,dimension);
```

```
276     temp_field_LUT = flip(temp_field);
277 else
278     temp_field = reset_temp_field(temp_field,1,dimension);
279     temp_field_LUT = flip(temp_field);
280 end
281
282 for k = 1:traj_raw_samples-2 %-2 to make even...
283     if static_or_dynamic == 1
284         if k >= re_slice
285             %disp('yes')
286             temp_field = update_temp_field(temp_field,0.100,
                    step,dimension);
287             temp_field_LUT = flip(temp_field);
288             step = step + step_increment;
289             re_slice = re_slice + s/temp_step_denom; %gives
                    ~32 temp steps...since -4 --> +4 with 0.25
                    step...
290         end
291     end
292     if k >= time_step %Simulation data is 5Hz...therefore
            seconds are every 5 data points...
293         second_tick = second_tick + 1;
294         for i = 1:33
295             for j = 1:33
296                 distance = sqrt((so_x_data(k)-
```

```
                    field_position_x(i))^2+(so_y_data(k)-
                    field_position_y(j))^2);
297                  if distance <= min_distance_value
298                      min_distance_value = distance;
299                      min_distance_x = i;
300                      min_distance_y = j;
301                  end
302              end
303          end
304      time_step = time_step + 5*speed_mult;
305      LUT_temp_value_0(second_tick) = temp_field_LUT(34-
         min_distance_y,min_distance_x);
306      if LUT_temp_value_0(second_tick) ~= LUT_temp_value_0(
         second_tick-1)
307          previous_LUT_temp_value = sensor_0_model_value(
             second_tick-1);
308          temp_time = 1; %reset
309          sensor_0_model_value(second_tick) = (
             LUT_temp_value_0(second_tick)-
             previous_LUT_temp_value)*(1-exp(-(temp_time)/
             model_tau))+previous_LUT_temp_value;
310      elseif LUT_temp_value_0(second_tick) == LUT_temp_value_0
         (second_tick-1)
311          sensor_0_model_value(second_tick) = (
             LUT_temp_value_0(second_tick)-
```

```
                previous_LUT_temp_value)*(1-exp(-(temp_time)/
                model_tau))+previous_LUT_temp_value;
312             temp_time = temp_time + 1;
313         end
314         fprintf("\n\nSeconds = %d\nSensor_x = %f\nSensor_y = %f\
              nMin-Dist = %f\nField-pos-x = %d\nField-pos-y = %d\
              nLUT-temp-val = %d\n\n",second_tick-1,so_x_data(k),
              so_y_data(k), min_distance_value,min_distance_x,
              min_distance_y,LUT_temp_value_o(second_tick));
315         min_distance_value = 100; %arbitrary upper reset...
316         end
317 end
318
319
320
321
322
323
324 figure('Name','Sensor-1 Response','NumberTitle','off')
325 plot(x_time_sec,sensor_3_model_value(x_time_sec));
326 title('Sensor - 1 Response')
327 legend('s1')
328 xlabel('time [sec]','FontSize',12)
329 xlim([1 floor(traj_raw_samples/(5*speed_mult))]) %188 flower
      , 110 lawn, 165 sprial, 86 star
```

```
330 ylabel('Temperature [^\circC]','FontSize',12)
331 if static_or_dynamic == 0
332     ylim([0.75 3.75])
333     yticks(0.75:.25:3.75)
334 else
335     ylim([0.75 3.75])
336     yticks(0.75:.25:3.75)
337 end
338 set(gcf,'color','w');
339
340 figure('Name','Sensor-2 Response','NumberTitle','off')
341 plot(x_time_sec,sensor_2_model_value(x_time_sec));
342 title('Sensor - 2 Response')
343 legend('s2')
344 xlabel('time [sec]','FontSize',12)
345 xlim([1 floor(traj_raw_samples/(5*speed_mult))])
346 ylabel('Temperature [^\circC]','FontSize',12)
347 if static_or_dynamic == 0
348     ylim([0.75 3.75])
349     yticks(0.75:.25:3.75)
350 else
351     ylim([0.75 3.75])
352     yticks(0.75:.25:3.75)
353 end
354 set(gcf,'color','w');
```

```
355
356 figure('Name','Sensor-3 Response','NumberTitle','off')
357 plot(x_time_sec,sensor_1_model_value(x_time_sec));
358 title('Sensor - 3 Response')
359 legend('s3')
360 xlabel('time [sec]','FontSize',12)
361 xlim([1 floor(traj_raw_samples/(5*speed_mult))])
362 ylabel('Temperature [^\circC]','FontSize',12)
363 if static_or_dynamic == 0
364     ylim([0.75 3.75])
365     yticks(0.75:.25:3.75)
366 else
367     ylim([0.75 3.75])
368     yticks(0.75:.25:3.75)
369 end
370 set(gcf,'color','w');
371
372 figure('Name','Sensor-4 Response','NumberTitle','off')
373 plot(x_time_sec,sensor_0_model_value(x_time_sec));
374 title('Sensor - 4 Response')
375 legend('s4')
376 xlabel('time [sec]','FontSize',12)
377 xlim([1 floor(traj_raw_samples/(5*speed_mult))])
378 ylabel('Temperature [^\circC]','FontSize',12)
379 if static_or_dynamic == 0
```

```
380         ylim([0.75  3.75])
381         yticks(0.75:.25:3.75)
382     else
383         ylim([0.75  3.75])
384         yticks(0.75:.25:3.75)
385     end
386     set(gcf,'color','w');
387
388     figure('Name','Sensor: 1 - 4 Response','NumberTitle','off')
389     plot(x_time_sec,sensor_0_model_value(x_time_sec),x_time_sec,
            sensor_1_model_value(x_time_sec),x_time_sec,
            sensor_2_model_value(x_time_sec),x_time_sec,
            sensor_3_model_value(x_time_sec));
390     title('Sensor: 1 - 4 Response')
391     legend('s4','s3','s2','s1')
392     xlabel('time [sec]','FontSize',12)
393     xlim([1  floor(traj_raw_samples/(5*speed_mult))])
394     ylabel('Temperature [^\circC]','FontSize',12)
395     if static_or_dynamic == 0
396         ylim([0.75  3.75])
397         yticks(0.75:.25:3.75)
398     else
399         ylim([0.75  3.75])
400         yticks(0.75:.25:3.75)
401     end
```

```
402 set(gcf,'color','w');
403 %
    %--------------------------------------------------------------||
```

B.8 Post-Processing: Temperature Field Reconstruction

```
clc
close all

recon_time = sim_seconds;

x_0 = so_x_data(:,1:5:end);
y_0 = so_y_data(:,1:5:end);
z_0 = so_z_data(:,1:5:end)*-1;
v_0 = sensor_0_model_value(:,1:1:end-1);

d_x_0 = -4:0.25:4;
d_y_0 = -4:0.25:4;
[xq_0,yq_0,zq_0] = meshgrid(d_x_0,d_y_0,14.79); %not sure
    whats up with the z values...ugh

set(0,'DefaultLegendAutoUpdate','off') %prevents 'data data
    data added to legend'

vq_0 = griddata(x_0(1:recon_time),y_0(1:recon_time),z_0(1:
    recon_time),v_0(1:recon_time),xq_0,yq_0,zq_0,'natural');
    %natural neighbor interpolation (could be linear, or
    nearest (gives full box))
fucks = 0;
for x = 1:33
```

```
20      for y = 1:33
21          if isnan(vq_0(x,y)) == 0
22              fucks = fucks + 1;
23          end
24      end
25  end
26  disp(fucks)
27  figure
28  po = plot3(x_0(1:recon_time),y_0(1:recon_time),z_0(1:
        recon_time),'k');
29  legend(po,{'s4'},'Location','Northeast','FontSize',12)
30  hold on
31  %title('Sensor - 4: Temperature Field Reconstruction','
        FontSize',12)
32  %legend('s4')
33  xlabel('Easting [m]','FontSize',12)
34  ylabel('Northing [m]','FontSize',12)
35  xlim([-4 4])
36  xticks(-4:1:4)
37  ylim([-4 4])
38  %zlim([-26 26])
39  shading flat
40  %cbar = colorbar('AxisLocation','out');
41  % ylabel(cbar, 'Temperature [\circC]')
42  cbar = colorbar('AxisLocation','out');
```

```
lim = caxis;
%caxis([0 3.1])
ylabel(cbar, 'Temperature [\circC]','FontSize',12)
%lim = caxis;
%caxis([1.75 3.50])
set(gcf,'color','w');
surf(xq_0,yq_0,vq_0);
view(2);
text(-3.6,-3.6,sprintf('t = %d [s]',recon_time),'color','k',
    'FontSize',12)
text(1.8,-3.6,'v = 0.5 [m/s]','color','k','FontSize',12)
axis square
hold off


x_1 = s1_x_data(:,1:floor(5*speed_mult):end-4);
y_1 = s1_y_data(:,1:floor(5*speed_mult):end-4);
z_1 = s1_z_data(:,1:floor(5*speed_mult):end-4)*-1;
v_1 = sensor_1_model_value(:,1:floor(traj_raw_samples/floor
    (5*speed_mult)));

d_x_1 = -4:0.25:4;
d_y_1 = -4:0.25:4;
[xq_1,yq_1,zq_1] = meshgrid(d_x_1,d_y_1,11.3);
```

```
66
67 vq_1 = griddata(x_1(1:recon_time),y_1(1:recon_time),z_1(1:
       recon_time),v_1(1:recon_time),xq_1,yq_1,zq_1,'natural');
      %natural neighbor interpolation (could be linear, or
      nearest)
68 figure
69 p1 = plot3(x_1(1:recon_time),y_1(1:recon_time),z_1(1:
       recon_time),'k');
70 legend(p1,{'s3'},'Location','Northeast','FontSize',12)
71 hold on
72 title('Sensor - 3: Temperature Field Reconstruction','
       FontSize',12)
73 %legend('s4')
74 xlabel('Easting [m]','FontSize',12)
75 ylabel('Northing [m]','FontSize',12)
76 xlim([-4 4])
77 xticks(-4:1:4)
78 ylim([-4 4])
79 shading flat
80 cbar = colorbar('AxisLocation','out');
81 ylabel(cbar, 'Temperature [\circC]','FontSize',12)
82 lim = caxis;
83 caxis([1.75 3.50])
84 set(gcf,'color','w');
85 surf(xq_1,yq_1,vq_1);
```

```
86 view(2);
87 text(-3.6,-3.6,sprintf('t = %d [s]',recon_time),'color','k',
       'FontSize',12)
88 text(1.8,-3.6,'v = 0.5 [m/s]','color','k','FontSize',12)
89 axis square
90 hold off
91
92
93
94 x_2 = s2_x_data(:,1:floor(5*speed_mult):end-4);
95 y_2 = s2_y_data(:,1:floor(5*speed_mult):end-4);
96 z_2 = s2_z_data(:,1:floor(5*speed_mult):end-4)*-1;
97 v_2 = sensor_2_model_value(:,1:floor(traj_raw_samples/floor
       (5*speed_mult)));
98
99 d_x_2 = -4:0.25:4;
100 d_y_2 = -4:0.25:4;
101 [xq_2,yq_2,zq_2] = meshgrid(d_x_2,d_y_2,7.8);
102
103 vq_2 = griddata(x_2(1:recon_time),y_2(1:recon_time),z_2(1:
       recon_time),v_2(1:recon_time),xq_2,yq_2,zq_2,'natural');
      %natural neighbor interpolation (could be linear, or
       nearest)
104 figure
105 p2 = plot3(x_2(1:recon_time),y_2(1:recon_time),z_2(1:
```

```
    recon_time),'k');
106 legend(p2,{'s2'},'Location','Northeast','FontSize',12)
107 hold on
108 title('Sensor - 2: Temperature Field Reconstruction','
    FontSize',12)
109 %legend('s4')
110 xlabel('Easting [m]','FontSize',12)
111 ylabel('Northing [m]','FontSize',12)
112 xlim([-4 4])
113 xticks(-4:1:4)
114 ylim([-4 4])
115 shading flat
116 cbar = colorbar('AxisLocation','out');
117 ylabel(cbar, 'Temperature [\circC]','FontSize',12)
118 lim = caxis;
119 caxis([1.75 3.50])
120 set(gcf,'color','w');
121 surf(xq_2,yq_2,vq_2);
122 view(2);
123 text(-3.6,-3.6,sprintf('t = %d [s]',recon_time),'color','k',
    'FontSize',12)
124 text(1.8,-3.6,'v = 0.5 [m/s]','color','k','FontSize',12)
125 axis square
126 hold off
127
```

```
128
129
130 x_3 = s3_x_data(:,1:floor(5*speed_mult):end-4);
131 y_3 = s3_y_data(:,1:floor(5*speed_mult):end-4);
132 z_3 = s3_z_data(:,1:floor(5*speed_mult):end-4)*-1;
133 v_3 = sensor_3_model_value(:,1:floor(traj_raw_samples/floor
      (5*speed_mult)));
134
135 d_x_3 = -4:0.25:4;
136 d_y_3 = -4:0.25:4;
137 [xq_3,yq_3,zq_3] = meshgrid(d_x_3,d_y_3,4.4);
138
139 vq_3 = griddata(x_3(1:recon_time),y_3(1:recon_time),z_3(1:
      recon_time),v_3(1:recon_time),xq_3,yq_3,zq_3,'natural');
      %natural neighbor interpolation (could be linear, or
      nearest)
140 figure
141 p3 = plot3(x_3(1:recon_time),y_3(1:recon_time),z_3(1:
      recon_time),'k');
142 legend(p3,{'s1'},'Location','Northeast','FontSize',12)
143 hold on
144 title('Sensor - 1: Temperature Field Reconstruction','
      FontSize',12)
145 %legend('s4')
146 xlabel('Easting [m]','FontSize',12)
```

```
147 ylabel('Northing [m]','FontSize',12)
148 xlim([-4 4])
149 xticks(-4:1:4)
150 ylim([-4 4])
151 shading flat
152 cbar = colorbar('AxisLocation','out');
153 ylabel(cbar, 'Temperature [\circC]','FontSize',12)
154 lim = caxis;
155 caxis([1.75 3.50])
156 set(gcf,'color','w');
157 surf(xq_3,yq_3,vq_3);
158 view(2);
159 text(-3.6,-3.6,sprintf('t = %d [s]',recon_time),'color','k',
      'FontSize',12)
160 text(1.8,-3.6,'v = 0.5 [m/s]','color','k','FontSize',12)
161 axis square
162 hold off
163
164 %error_analysis
```

B.9 Post-Processing: Reconstruction Error Analysis

```
clc
close all

so_detect_time = 0;
for q = 1:traj_raw_samples/floor(5*speed_mult)
    if sensor_0_model_value(q) == 3.50
        so_detect_time = so_detect_time + 1;
    else

    end
end
%disp(so_detect_time);

so_input_field_values = 0;
so_input_field_min_val = 100;
so_input_field_max_val = 0;


flipped_temp_field = flip(temp_field);
flipped_vq_0 = flip(vq_0,1);
so_sum_diff = 0;
so_num_val = 0;
so_values = 0;
so_min_val = 100; %arbitrary high
```

```
25 so_max_val = 0; %arbitrary low
26 for x = 1:33
27     for y = 1:33
28         if isnan(flipped_vq_o(x,y))
29         else
30             if flipped_vq_o(x,y) <= so_min_val
31                 so_min_val = flipped_vq_o(x,y);
32             end
33             if flipped_temp_field(x,y,1) <=
                   so_input_field_min_val
34                 so_input_field_min_val = flipped_temp_field(
                       x,y,1);
35             end
36             if flipped_vq_o(x,y) >= so_max_val
37                 so_max_val = flipped_vq_o(x,y);
38             end
39             if flipped_temp_field(x,y,1) >=
                   so_input_field_max_val
40                 so_input_field_max_val = flipped_temp_field(
                       x,y,1);
41             end
42             so_input_field_values = so_input_field_values +
                   flipped_temp_field(x,y,1);
43             so_values = so_values + flipped_vq_o(x,y);
44             so_num_val = so_num_val + 1;
```

```
45              so_sum_diff = so_sum_diff + (flipped_temp_field(
                    x,y,1)-flipped_vq_o(x,y))^2;
46              %fprintf("x = %d\ny = %d\ntemp_field = %f\nvq_o
                    = %f\n\n",x,y,flipped_temp_field(x,y,1),
                    flipped_vq_o(x,y));
47          end
48      end
49  end
50
51  so_rmse = sqrt(so_sum_diff/so_num_val);
52  so_avg_val = (so_values/so_num_val);
53  so_sum_var_diff = 0;
54  so_input_avg_val = (so_input_field_values/so_num_val);
55  so_input_sum_var_diff = 0;
56
57  so_median_mode_array = ones(size(1:so_num_val));
58  so_input_median_mode_array = ones(size(1:so_num_val));
59  median_mode_step = 1;
60
61  for x = 1:33
62      for y = 1:33
63          if isnan(flipped_vq_o(x,y))
64          else
65              so_median_mode_array(median_mode_step) =
                    flipped_vq_o(x,y);
```

```
66                  so_input_median_mode_array(median_mode_step) =
                        flipped_temp_field(x,y,1);
67                  median_mode_step = median_mode_step + 1;
68                  so_sum_var_diff = so_sum_var_diff + ((
                        flipped_vq_o(x,y)-so_avg_val))^2;
69                  so_input_sum_var_diff = so_input_sum_var_diff +
                        ((flipped_temp_field(x,y,1)-so_input_avg_val)
                        )^2;
70              end
71          end
72      end
73
74      so_var = (so_sum_var_diff/(so_num_val-1));
75      so_std = sqrt(so_var);
76      so_range = so_max_val - so_min_val;
77      so_median = median(so_median_mode_array);
78      so_mode = mode(so_median_mode_array);
79
80      so_input_var = (so_input_sum_var_diff/(so_num_val-1));
81      so_input_std = sqrt(so_input_var);
82      so_input_median = median(so_input_median_mode_array);
83      so_input_mode = mode(so_input_median_mode_array);
84      so_input_range = so_input_field_max_val -
            so_input_field_min_val;
85
```

```
86 fprintf("----------\nso_input_avg_val =\t%0.3f [\260C]\
     nso_input_var =\t\t%0.3f [\260C]\nso_input_std =\t\t%0.3f
      [\260C]\nso_input_median =\t%0.3f [\260C]\nso_input_mode
      =\t\t%0.3f [\260C]\nso_input_min_val =\t%0.3f [\260C]\
     nso_input_max_val =\t%0.3f [\260C]\nso_input_range =\t
     %0.3f [\260C]\n\nso_detect_time =\t%d [sec]\nso_num_val
     =\t\t%d [#]\nso_rmse =\t\t\t%0.3f [\260C]\n\nso_avg_val
     =\t\t%0.3f [\260C]\nso_var = \t\t\t%0.3f [\260C^2]\
     nso_std = \t\t\t%0.3f [\260C]\nso_median = \t\t%0.3f
     [\260C]\nso_mode = \t\t\t%0.3f [\260C]\nso_min_val =\t\t
     %0.3f [\260C]\nso_max_val =\t\t%0.3f [\260C]\nso_range =\
     t\t\t%0.3f [\260C]\n----------\n\n",so_input_avg_val,
     so_input_var,so_input_std,so_input_median,so_input_mode,
     so_input_field_min_val,so_input_field_max_val,
     so_input_range,so_detect_time,so_num_val,so_rmse,
     so_avg_val,so_var,so_std,so_median,so_mode,so_min_val,
     so_max_val,so_range);
87 so_fileID = fopen('so_analysis.txt','w');
88 fprintf(so_fileID,"----------\nso_input_avg_val =\t%0.3f
     [\260C]\nso_input_var =\t\t%0.3f [\260C]\nso_input_std =\
     t\t%0.3f [\260C]\nso_input_median =\t%0.3f [\260C]\
     nso_input_mode =\t\t%0.3f [\260C]\nso_input_min_val =\t
     %0.3f [\260C]\nso_input_max_val =\t%0.3f [\260C]\
     nso_input_range =\t%0.3f [\260C]\n\nso_detect_time =\t%d
     [sec]\nso_num_val =\t\t%d [#]\nso_rmse =\t\t%0.3f [\260C
```

```
    ]\n\nso_avg_val =\t\t%0.3f [\260C]\nso_var =\t\t%0.3f
    [\260C^2]\nso_std =\t\t%0.3f [\260C]\nso_median = \t\t
    %0.3f [\260C]\nso_mode =\t\t%0.3f [\260C]\nso_min_val =\t
    \t%0.3f [\260C]\nso_max_val =\t\t%0.3f [\260C]\nso_range
    =\t\t%0.3f [\260C]\n----------\n\n",so_input_avg_val,
    so_input_var,so_input_std,so_input_median,so_input_mode,
    so_input_field_min_val,so_input_field_max_val,
    so_input_range,so_detect_time,so_num_val,so_rmse,
    so_avg_val,so_var,so_std,so_median,so_mode,so_min_val,
    so_max_val,so_range);
fclose(so_fileID);



s1_detect_time = 0;
for q = 1:traj_raw_samples/floor(5*speed_mult)
    if sensor_1_model_value(q) == 3.50
        s1_detect_time = s1_detect_time + 1;
    else

    end
end
%disp(s1_detect_time);

```

```
104 s1_input_field_values = 0;
105 s1_input_field_min_val = 100;
106 s1_input_field_max_val = 0;
107
108
109 flipped_temp_field = flip(temp_field);
110 flipped_vq_1 = flip(vq_1,1);
111 s1_sum_diff = 0;
112 s1_num_val = 0;
113 s1_values = 0;
114 s1_min_val = 100; %arbitrary high
115 s1_max_val = 0; %arbitrary low
116 for x = 1:33
117     for y = 1:33
118         if isnan(flipped_vq_1(x,y))
119         else
120             if flipped_vq_1(x,y) <= s1_min_val
121                 s1_min_val = flipped_vq_1(x,y);
122             end
123             if flipped_temp_field(x,y,1) <=
                    s1_input_field_min_val
124                 s1_input_field_min_val = flipped_temp_field(
                        x,y,1);
125             end
126             if flipped_vq_1(x,y) >= s1_max_val
```

```
127                     s1_max_val = flipped_vq_1(x,y);
128                 end
129                 if flipped_temp_field(x,y,1) >=
                        s1_input_field_max_val
130                     s1_input_field_max_val = flipped_temp_field(
                            x,y,1);
131                 end
132                 s1_input_field_values = s1_input_field_values +
                        flipped_temp_field(x,y,1);
133                 s1_values = s1_values + flipped_vq_1(x,y);
134                 s1_num_val = s1_num_val + 1;
135                 s1_sum_diff = s1_sum_diff + (flipped_temp_field(
                        x,y,1)-flipped_vq_1(x,y))^2;
136                 %fprintf("x = %d\ny = %d\ntemp_field = %f\nvq_o
                        = %f\n\n",x,y,flipped_temp_field(x,y,1),
                        flipped_vq_o(x,y));
137             end
138         end
139 end
140
141 s1_rmse = sqrt(s1_sum_diff/s1_num_val);
142 s1_avg_val = (s1_values/s1_num_val);
143 s1_sum_var_diff = 0;
144 s1_input_avg_val = (s1_input_field_values/s1_num_val);
145 s1_input_sum_var_diff = 0;
```

```
146
147 s1_median_mode_array = ones(size(1:s1_num_val));
148 s1_input_median_mode_array = ones(size(1:s1_num_val));
149 median_mode_step = 1;
150
151 for x = 1:33
152     for y = 1:33
153         if isnan(flipped_vq_1(x,y))
154         else
155             s1_median_mode_array(median_mode_step) =
                flipped_vq_1(x,y);
156             s1_input_median_mode_array(median_mode_step) =
                flipped_temp_field(x,y,1);
157             median_mode_step = median_mode_step + 1;
158             s1_sum_var_diff = s1_sum_var_diff + ((
                flipped_vq_1(x,y)-s1_avg_val))^2;
159             s1_input_sum_var_diff = s1_input_sum_var_diff +
                ((flipped_temp_field(x,y,1)-s1_input_avg_val)
                )^2;
160         end
161     end
162 end
163
164 s1_var = (s1_sum_var_diff/(s1_num_val-1));
165 s1_std = sqrt(s1_var);
```

```
166 s1_range = s1_max_val - s1_min_val;
167 s1_median = median(s1_median_mode_array);
168 s1_mode = mode(s1_median_mode_array);
169
170 s1_input_var = (s1_input_sum_var_diff/(s1_num_val-1));
171 s1_input_std = sqrt(s1_input_var);
172 s1_input_median = median(s1_input_median_mode_array);
173 s1_input_mode = mode(s1_input_median_mode_array);
174 s1_input_range = s1_input_field_max_val -
      s1_input_field_min_val;
175
176 fprintf("----------\ns1_input_avg_val =\t%0.3f [\260C]\
    ns1_input_var =\t\t%0.3f [\260C]\ns1_input_std =\t\t%0.3f
     [\260C]\ns1_input_median =\t%0.3f [\260C]\ns1_input_mode
     =\t\t%0.3f [\260C]\ns1_input_min_val =\t%0.3f [\260C]\
    ns1_input_max_val =\t%0.3f [\260C]\ns1_input_range =\t
    %0.3f [\260C]\n\ns1_detect_time =\t%d [sec]\ns1_num_val
    =\t\t%d [#]\ns1_rmse =\t\t\t%0.3f [\260C]\n\ns1_avg_val
    =\t\t%0.3f [\260C]\ns1_var = \t\t\t%0.3f [\260C^2]\
    ns1_std = \t\t\t%0.3f [\260C]\ns1_median = \t\t%0.3f
    [\260C]\ns1_mode = \t\t\t%0.3f [\260C]\ns1_min_val =\t\t
    %0.3f [\260C]\ns1_max_val =\t\t%0.3f [\260C]\ns1_range =\
    t\t\t%0.3f [\260C]\n----------\n\n",s1_input_avg_val,
    s1_input_var,s1_input_std,s1_input_median,s1_input_mode,
    s1_input_field_min_val,s1_input_field_max_val,
```

```
    s1_input_range,s1_detect_time,s1_num_val,s1_rmse,
    s1_avg_val,s1_var,s1_std,s1_median,s1_mode,s1_min_val,
    s1_max_val,s1_range);
177 s1_fileID = fopen('s1_analysis.txt','w');
178 fprintf(s1_fileID,"----------\ns1_input_avg_val =\t%0.3f
    [\260C]\ns1_input_var =\t\t%0.3f [\260C]\ns1_input_std =\
    t\t%0.3f [\260C]\ns1_input_median =\t%0.3f [\260C]\
    ns1_input_mode =\t\t%0.3f [\260C]\ns1_input_min_val =\t
    %0.3f [\260C]\ns1_input_max_val =\t%0.3f [\260C]\
    ns1_input_range =\t%0.3f [\260C]\n\ns1_detect_time =\t%d
    [sec]\ns1_num_val =\t\t%d [#]\ns1_rmse =\t\t%0.3f [\260C
    ]\n\ns1_avg_val =\t\t%0.3f [\260C]\ns1_var =\t\t%0.3f
    [\260C^2]\ns1_std =\t\t%0.3f [\260C]\ns1_median = \t\t
    %0.3f [\260C]\ns1_mode =\t\t%0.3f [\260C]\ns1_min_val =\t
    \t%0.3f [\260C]\ns1_max_val =\t\t%0.3f [\260C]\ns1_range
    =\t\t%0.3f [\260C]\n----------\n\n",s1_input_avg_val,
    s1_input_var,s1_input_std,s1_input_median,s1_input_mode,
    s1_input_field_min_val,s1_input_field_max_val,
    s1_input_range,s1_detect_time,s1_num_val,s1_rmse,
    s1_avg_val,s1_var,s1_std,s1_median,s1_mode,s1_min_val,
    s1_max_val,s1_range);
179 fclose(s1_fileID);
180
181
182
```

```
183
184 s2_detect_time = 0;
185 for q = 1:traj_raw_samples/floor(5*speed_mult)
186     if sensor_2_model_value(q) == 3.50
187         s2_detect_time = s2_detect_time + 1;
188     else
189
190     end
191 end
192 %disp(s2_detect_time);
193
194 s2_input_field_values = 0;
195 s2_input_field_min_val = 100;
196 s2_input_field_max_val = 0;
197
198
199 flipped_temp_field = flip(temp_field);
200 flipped_vq_2 = flip(vq_2,1);
201 s2_sum_diff = 0;
202 s2_num_val = 0;
203 s2_values = 0;
204 s2_min_val = 100; %arbitrary high
205 s2_max_val = 0; %arbitrary low
206 for x = 1:33
207     for y = 1:33
```

```
208         if isnan(flipped_vq_2(x,y))
209         else
210             if flipped_vq_2(x,y) <= s2_min_val
211                 s2_min_val = flipped_vq_2(x,y);
212             end
213             if flipped_temp_field(x,y,1) <=
                    s2_input_field_min_val
214                 s2_input_field_min_val = flipped_temp_field(
                        x,y,1);
215             end
216             if flipped_vq_2(x,y) >= s2_max_val
217                 s2_max_val = flipped_vq_2(x,y);
218             end
219             if flipped_temp_field(x,y,1) >=
                    s2_input_field_max_val
220                 s2_input_field_max_val = flipped_temp_field(
                        x,y,1);
221             end
222             s2_input_field_values = s2_input_field_values +
                    flipped_temp_field(x,y,1);
223             s2_values = s2_values + flipped_vq_2(x,y);
224             s2_num_val = s2_num_val + 1;
225             s2_sum_diff = s2_sum_diff + (flipped_temp_field(
                    x,y,1)-flipped_vq_2(x,y))^2;
226            %fprintf("x = %d\ny = %d\ntemp_field = %f\nvq_o
```

```
                = %f\n\n",x,y,flipped_temp_field(x,y,1),
                flipped_vq_0(x,y));
227         end
228     end
229 end
230
231 s2_rmse = sqrt(s2_sum_diff/s2_num_val);
232 s2_avg_val = (s2_values/s2_num_val);
233 s2_sum_var_diff = 0;
234 s2_input_avg_val = (s2_input_field_values/s2_num_val);
235 s2_input_sum_var_diff = 0;
236
237 s2_median_mode_array = ones(size(1:s2_num_val));
238 s2_input_median_mode_array = ones(size(1:s2_num_val));
239 median_mode_step = 1;
240
241 for x = 1:33
242     for y = 1:33
243         if isnan(flipped_vq_2(x,y))
244         else
245             s2_median_mode_array(median_mode_step) =
                flipped_vq_2(x,y);
246             s2_input_median_mode_array(median_mode_step) =
                flipped_temp_field(x,y,1);
247             median_mode_step = median_mode_step + 1;
```

```
248                 s2_sum_var_diff = s2_sum_var_diff + ((
                        flipped_vq_2(x,y)-s2_avg_val))^2;
249                 s2_input_sum_var_diff = s2_input_sum_var_diff +
                        ((flipped_temp_field(x,y,1)-s2_input_avg_val)
                        )^2;
250             end
251         end
252 end
253
254 s2_var = (s2_sum_var_diff/(s2_num_val-1));
255 s2_std = sqrt(s2_var);
256 s2_range = s2_max_val - s2_min_val;
257 s2_median = median(s2_median_mode_array);
258 s2_mode = mode(s2_median_mode_array);
259
260 s2_input_var = (s2_input_sum_var_diff/(s2_num_val-1));
261 s2_input_std = sqrt(s2_input_var);
262 s2_input_median = median(s2_input_median_mode_array);
263 s2_input_mode = mode(s2_input_median_mode_array);
264 s2_input_range = s2_input_field_max_val -
        s2_input_field_min_val;
265
266 fprintf("----------\ns2_input_avg_val =\t%0.3f [\260C]\
        ns2_input_var =\t\t%0.3f [\260C]\ns2_input_std =\t\t%0.3f
         [\260C]\ns2_input_median =\t%0.3f [\260C]\ns2_input_mode
```

```
 =\t\t%0.3f [\260C]\ns2_input_min_val =\t%0.3f [\260C]\
ns2_input_max_val =\t%0.3f [\260C]\ns2_input_range =\t
%0.3f [\260C]\n\ns2_detect_time =\t%d [sec]\ns2_num_val
=\t\t%d [#]\ns2_rmse =\t\t\t%0.3f [\260C]\n\ns2_avg_val
=\t\t%0.3f [\260C]\ns2_var = \t\t\t%0.3f [\260C^2]\
ns2_std = \t\t\t%0.3f [\260C]\ns2_median = \t\t%0.3f
[\260C]\ns2_mode = \t\t\t%0.3f [\260C]\ns2_min_val =\t\t
%0.3f [\260C]\ns2_max_val =\t\t%0.3f [\260C]\ns2_range =\
t\t\t%0.3f [\260C]\n----------\n\n",s2_input_avg_val,
s2_input_var,s2_input_std,s2_input_median,s2_input_mode,
s2_input_field_min_val,s2_input_field_max_val,
s2_input_range,s2_detect_time,s2_num_val,s2_rmse,
s2_avg_val,s2_var,s2_std,s2_median,s2_mode,s2_min_val,
s2_max_val,s2_range);
s2_fileID = fopen('s2_analysis.txt','w');
fprintf(s2_fileID,"----------\ns2_input_avg_val =\t%0.3f
[\260C]\ns2_input_var =\t\t%0.3f [\260C]\ns2_input_std =\
t\t%0.3f [\260C]\ns2_input_median =\t%0.3f [\260C]\
ns2_input_mode =\t\t%0.3f [\260C]\ns2_input_min_val =\t
%0.3f [\260C]\ns2_input_max_val =\t%0.3f [\260C]\
ns2_input_range =\t%0.3f [\260C]\n\ns2_detect_time =\t%d
[sec]\ns2_num_val =\t\t%d [#]\ns2_rmse =\t\t%0.3f [\260C
]\n\ns2_avg_val =\t\t%0.3f [\260C]\ns2_var =\t\t%0.3f
[\260C^2]\ns2_std =\t\t%0.3f [\260C]\ns2_median = \t\t
%0.3f [\260C]\ns2_mode =\t\t%0.3f [\260C]\ns2_min_val =\t
```

```
    \t%0.3f [\260C]\ns2_max_val =\t\t%0.3f [\260C]\ns2_range
    =\t\t%0.3f [\260C]\n----------\n\n",s2_input_avg_val,
    s2_input_var,s2_input_std,s2_input_median,s2_input_mode,
    s2_input_field_min_val,s2_input_field_max_val,
    s2_input_range,s2_detect_time,s2_num_val,s2_rmse,
    s2_avg_val,s2_var,s2_std,s2_median,s2_mode,s2_min_val,
    s2_max_val,s2_range);
fclose(s2_fileID);



s3_detect_time = 0;
for q = 1:traj_raw_samples/floor(5*speed_mult)
    if sensor_3_model_value(q) == 3.50
        s3_detect_time = s3_detect_time + 1;
    else

    end
end
%disp(s3_detect_time);

s3_input_field_values = 0;
s3_input_field_min_val = 100;
```

```
287 s3_input_field_max_val = 0;
288
289
290 flipped_temp_field = flip(temp_field);
291 flipped_vq_3 = flip(vq_3,1);
292 s3_sum_diff = 0;
293 s3_num_val = 0;
294 s3_values = 0;
295 s3_min_val = 100; %arbitrary high
296 s3_max_val = 0; %arbitrary low
297 for x = 1:33
298     for y = 1:33
299         if isnan(flipped_vq_3(x,y))
300         else
301             if flipped_vq_3(x,y) <= s3_min_val
302                 s3_min_val = flipped_vq_3(x,y);
303             end
304             if flipped_temp_field(x,y,1) <=
                   s3_input_field_min_val
305                 s3_input_field_min_val = flipped_temp_field(
                       x,y,1);
306             end
307             if flipped_vq_3(x,y) >= s3_max_val
308                 s3_max_val = flipped_vq_3(x,y);
309             end
```

```
310                 if flipped_temp_field(x,y,1) >=
                        s3_input_field_max_val
311                     s3_input_field_max_val = flipped_temp_field(
                            x,y,1);
312                 end
313                 s3_input_field_values = s3_input_field_values +
                        flipped_temp_field(x,y,1);
314                 s3_values = s3_values + flipped_vq_3(x,y);
315                 s3_num_val = s3_num_val + 1;
316                 s3_sum_diff = s3_sum_diff + (flipped_temp_field(
                        x,y,1)-flipped_vq_3(x,y))^2;
317                 %fprintf("x = %d\ny = %d\ntemp_field = %f\nvq_0
                        = %f\n\n",x,y,flipped_temp_field(x,y,1),
                        flipped_vq_0(x,y));
318             end
319         end
320 end
321
322 s3_rmse = sqrt(s3_sum_diff/s3_num_val);
323 s3_avg_val = (s3_values/s3_num_val);
324 s3_sum_var_diff = 0;
325 s3_input_avg_val = (s3_input_field_values/s3_num_val);
326 s3_input_sum_var_diff = 0;
327
328 s3_median_mode_array = ones(size(1:s3_num_val));
```

```
329 s3_input_median_mode_array = ones(size(1:s3_num_val));
330 median_mode_step = 1;
331
332 for x = 1:33
333     for y = 1:33
334         if isnan(flipped_vq_3(x,y))
335         else
336             s3_median_mode_array(median_mode_step) =
                    flipped_vq_3(x,y);
337             s3_input_median_mode_array(median_mode_step) =
                    flipped_temp_field(x,y,1);
338             median_mode_step = median_mode_step + 1;
339             s3_sum_var_diff = s3_sum_var_diff + ((
                    flipped_vq_3(x,y)-s3_avg_val))^2;
340             s3_input_sum_var_diff = s3_input_sum_var_diff +
                    ((flipped_temp_field(x,y,1)-s3_input_avg_val)
                    )^2;
341         end
342     end
343 end
344
345 s3_var = (s3_sum_var_diff/(s3_num_val-1));
346 s3_std = sqrt(s3_var);
347 s3_range = s3_max_val - s3_min_val;
348 s3_median = median(s3_median_mode_array);
```

```
349 s3_mode = mode(s3_median_mode_array);
350
351 s3_input_var = (s3_input_sum_var_diff/(s3_num_val-1));
352 s3_input_std = sqrt(s3_input_var);
353 s3_input_median = median(s3_input_median_mode_array);
354 s3_input_mode = mode(s3_input_median_mode_array);
355 s3_input_range = s3_input_field_max_val -
      s3_input_field_min_val;
356
357 fprintf("----------\ns3_input_avg_val =\t%0.3f [\260C]\
    ns3_input_var =\t\t%0.3f [\260C]\ns3_input_std =\t\t%0.3f
     [\260C]\ns3_input_median =\t%0.3f [\260C]\ns3_input_mode
     =\t\t%0.3f [\260C]\ns3_input_min_val =\t%0.3f [\260C]\
    ns3_input_max_val =\t%0.3f [\260C]\ns3_input_range =\t
    %0.3f [\260C]\n\ns3_detect_time =\t%d [sec]\ns3_num_val
    =\t\t%d [#]\ns3_rmse =\t\t\t%0.3f [\260C]\n\ns3_avg_val
    =\t\t%0.3f [\260C]\ns3_var = \t\t\t%0.3f [\260C^2]\
    ns3_std = \t\t\t%0.3f [\260C]\ns3_median = \t\t%0.3f
    [\260C]\ns3_mode = \t\t\t%0.3f [\260C]\ns3_min_val =\t\t
    %0.3f [\260C]\ns3_max_val =\t\t%0.3f [\260C]\ns3_range =\
    t\t\t%0.3f [\260C]\n----------\n\n",s3_input_avg_val,
    s3_input_var,s3_input_std,s3_input_median,s3_input_mode,
    s3_input_field_min_val,s3_input_field_max_val,
    s3_input_range,s3_detect_time,s3_num_val,s3_rmse,
    s3_avg_val,s3_var,s3_std,s3_median,s3_mode,s3_min_val,
```

```
    s3_max_val,s3_range);
358 s3_fileID = fopen('s3_analysis.txt','w');
359 fprintf(s3_fileID,"----------\ns3_input_avg_val =\t%0.3f
    [\260C]\ns3_input_var =\t\t%0.3f [\260C]\ns3_input_std =\
    t\t%0.3f [\260C]\ns3_input_median =\t%0.3f [\260C]\
    ns3_input_mode =\t\t%0.3f [\260C]\ns3_input_min_val =\t
    %0.3f [\260C]\ns3_input_max_val =\t%0.3f [\260C]\
    ns3_input_range =\t%0.3f [\260C]\n\ns3_detect_time =\t%d
    [sec]\ns3_num_val =\t\t%d [#]\ns3_rmse =\t\t%0.3f [\260C
    ]\n\ns3_avg_val =\t\t%0.3f [\260C]\ns3_var =\t\t%0.3f
    [\260C^2]\ns3_std =\t\t%0.3f [\260C]\ns3_median = \t\t
    %0.3f [\260C]\ns3_mode =\t\t%0.3f [\260C]\ns3_min_val =\t
    \t%0.3f [\260C]\ns3_max_val =\t\t%0.3f [\260C]\ns3_range
    =\t\t%0.3f [\260C]\n----------\n\n",s3_input_avg_val,
    s3_input_var,s3_input_std,s3_input_median,s3_input_mode,
    s3_input_field_min_val,s3_input_field_max_val,
    s3_input_range,s3_detect_time,s3_num_val,s3_rmse,
    s3_avg_val,s3_var,s3_std,s3_median,s3_mode,s3_min_val,
    s3_max_val,s3_range);
360 fclose(s3_fileID);
```

B.10 Post-Processing: Reconstruction Difference to Input Field

```
close all
clc

value = zeros*meshgrid(1:1:33,1:1:33);

for x = 1:33
    for y = 1:33
        if isnan(flipped_vq_o(x,y))
        else
            value(x,y) = abs(flipped_temp_field(x,y,1)-
                flipped_vq_o(x,y));
            %disp(value(x,y))
            %fprintf("x = %d\ny = %d\ntemp_field = %f\nvq_o
                = %f\n\n",x,y,flipped_temp_field(x,y,1),
                flipped_vq_o(x,y));
        end
    end
end

disp(value)
surf(value)
view(0,-90)
xlabel('Column','FontSize',12)
ylabel('Row','FontSize',12)
```

```
22 cbar = colorbar('AxisLocation','out');
23 lim = caxis;
24 set(gcf,'color','w');
25 ylabel(cbar, 'Temperature [\circC]','FontSize',12)
26 text(2.6,30.6,sprintf('t = %d [s]',recon_time),'color','w','
     FontSize',12)
27
28 %
29 % d_x_0 = -4:0.25:4;
30 % d_y_0 = -4:0.25:4;
31 % [xq_0,yq_0,zq_0] = meshgrid(d_x_0,d_y_0,14.79); %not sure
     whats up with the z values...ugh
32 %
33 % vq_0 = griddata(x_0(1:75),y_0(1:75),z_0(1:75),v_0(1:75),
     xq_0,yq_0,zq_0,'linear'); %natural neighbor interpolation
      (could be linear, or nearest (gives full box))
34 % fucks = 0;
35 % for x = 1:33
36 %     for y = 1:33
37 %         if isnan(vq_0(x,y)) == 0
38 %             fucks = fucks + 1;
39 %         end
40 %     end
41 % end
42 % disp(fucks)
```

```
43 % figure
44 % %lot3(x_0(1:75),y_0(1:75),z_0(1:75),'k')
45 % hold on
46 % xlabel('Easting [m]')
47 % ylabel('Northing [m]')
48 % xlim([-4 4])
49 % xticks(-4:1:4)
50 % ylim([-4 4])
51 % shading flat
52 % cbar = colorbar('AxisLocation','out');
53 % lim = caxis;
54 % ylabel(cbar, 'Temperature [\circC]')
55 % set(gcf,'color','w');
56 % surf(xq_0,yq_0,vq_0);
57 % text(-3.6,-3.6,'t = 75 [s]','color','k','FontSize',12)
58 % axis square
59 % hold off
```

Appendix C

Mechanical Designs:

C.1 TAUS-Alpha Telescopic Mount

Figure C.1: A telescopic mount for the TAUS-Alpha Holy Stone micro-UAS. This was developed and integrated so we can test various power demands based on actual throttle, roll, pitch, and yaw commands.

C.2 DC Motor Mount

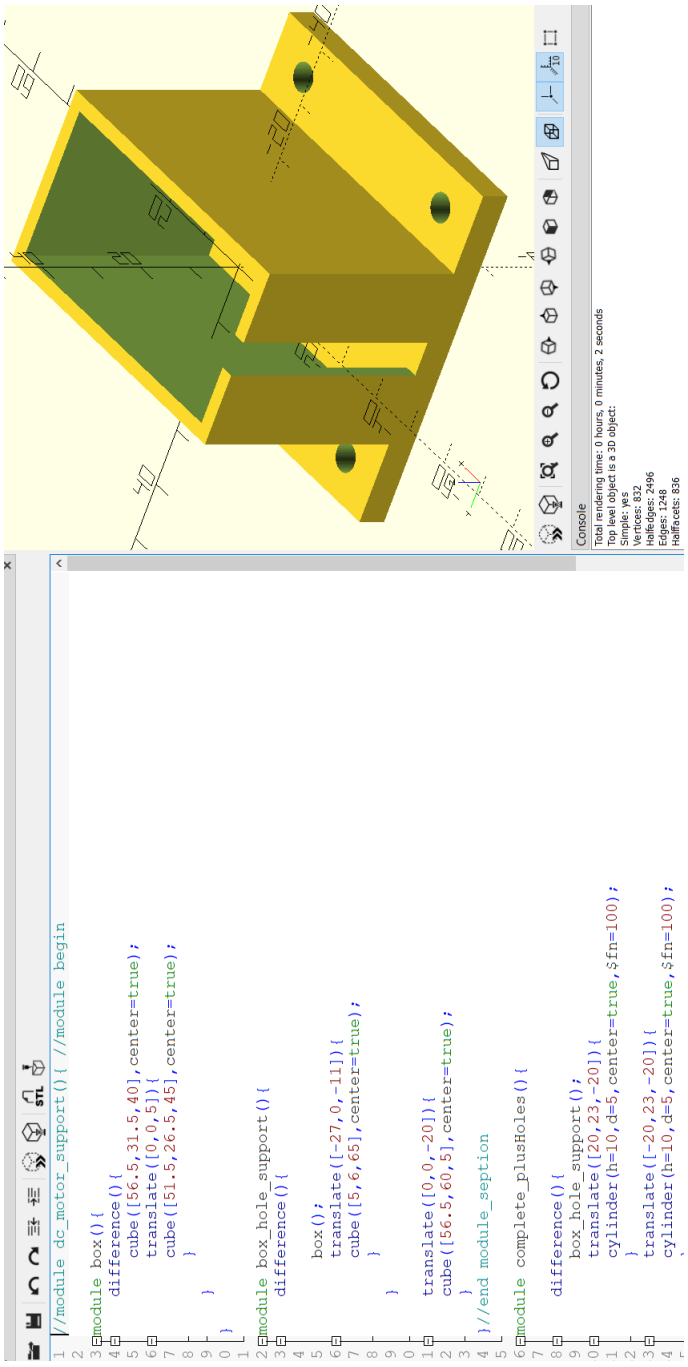
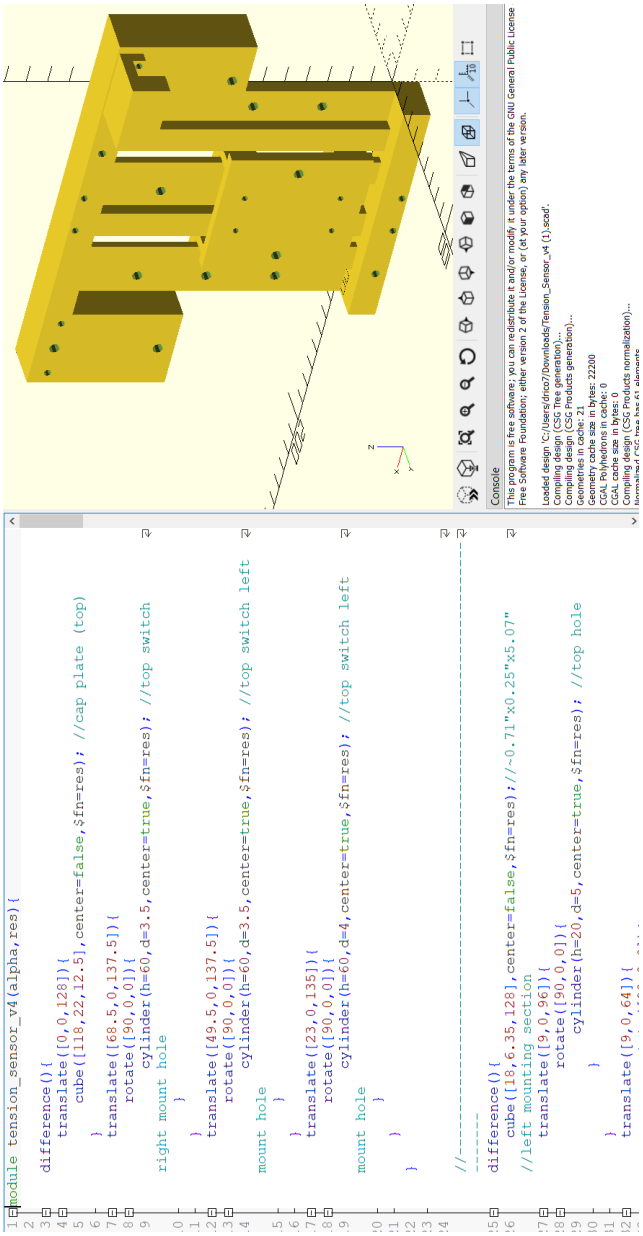


Figure C.2: The spool was actuated by a high-torque DC motor that we mounted inside the ground station with this structure.

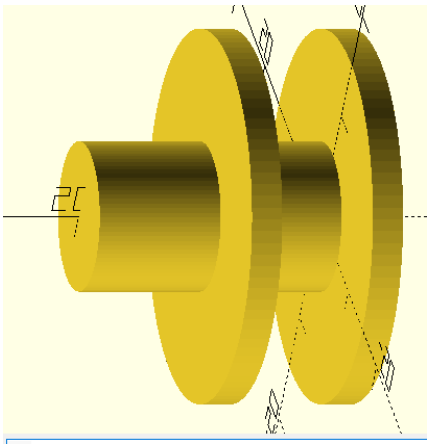
C.3 Spool Tension Feedback Mechanism



```
module tension_sensor_v4(alpha,res){
    difference(){
        translate([0,0,128]){
            cube([118,22,12.5],center=false,$fn=res); //cap plate (top)
        }
        translate([68.5,0,137.5]){
            rotate([90,0,0]){
                cylinder(h=60,d=3.5,center=true,$fn=res); //top switch
right mount hole
            }
        }
        translate([49.5,0,137.5]){
            rotate([90,0,0]){
                cylinder(h=60,d=3.5,center=true,$fn=res); //top switch left
mount hole
            }
        }
        translate([23,0,135]){
            rotate([90,0,0]){
                cylinder(h=60,d=4,center=true,$fn=res); //top switch left
mount hole
            }
        }
    }

//------
------
difference(){
    cube([18,6.35,128],center=false,$fn=res);//~0.71"x0.25"x5.07"
    //left mounting section
    translate([9,0,96]){
        rotate([90,0,0]){
            cylinder(h=20,d=5,center=true,$fn=res); //top hole
        }
    }
    translate([9,0,64]){
```

Figure C.3: To control the retract and release actuation of the power-sensing-tether spool we developed this structure to mount in the ground station. There are mounting opportunities for bearings and the middle section is free to actuate in the vertical as a result of tension on the tether.



```
module tension_sensor_wheel(res){
    translate([0,0,7.5]){
        cylinder(h=2.5,d=30,center=false,$fn=res); //top plate
    }
    cylinder(h=20,d=12,center=false,$fn=res); //column
    translate([0,0,-2.5]){
        cylinder(h=2.5,d=30,center=false,$fn=res); //bottom plate
    }
}//end module
tension_sensor_wheel(100);
```

Figure C.4: The bearings of the Spool Tension Feedback Mechanism mechanically clamped to these small roller spools to guide the tether through the mechanism.

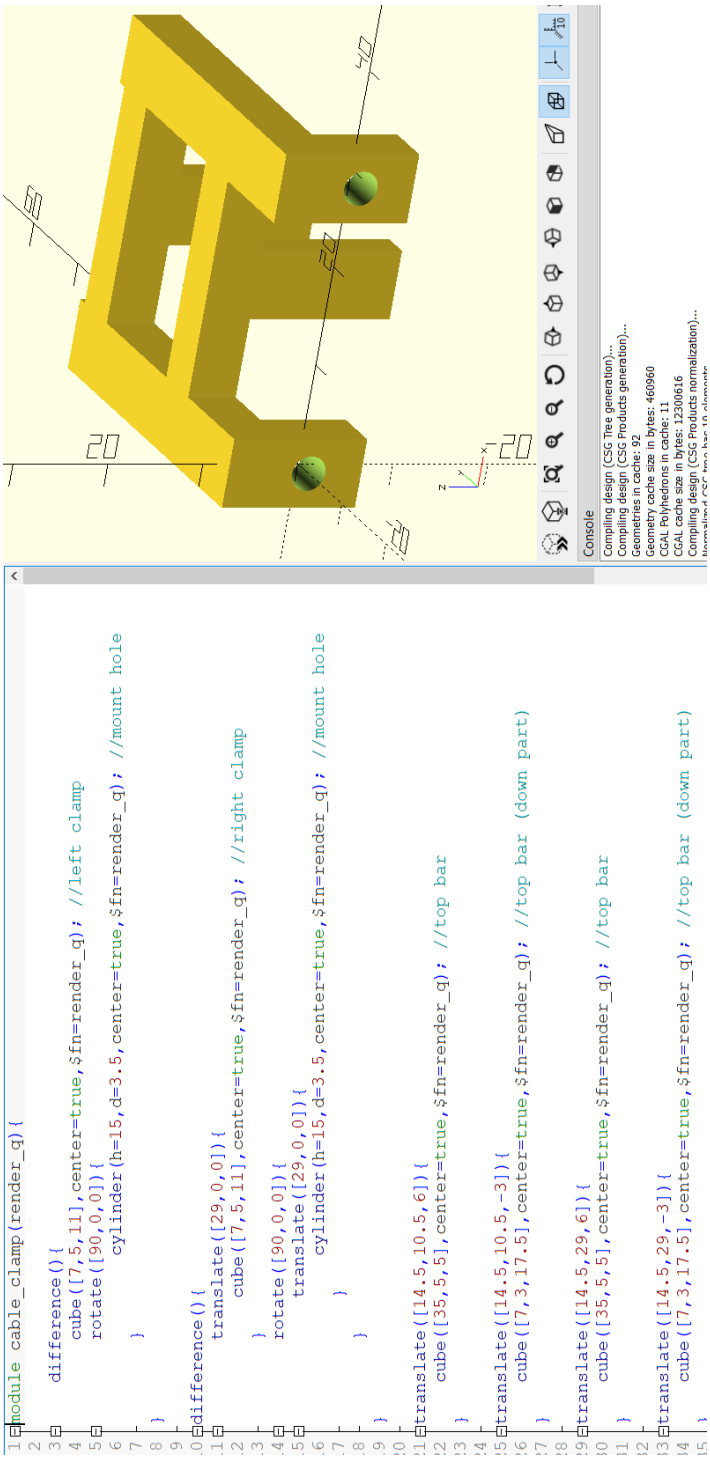


Figure C.5: To keep the tether contained and running along the small roller spools of the mechanism these clamps were needed. They acted as a barrier around the small spools to keep the tether in the run.

C.4 Linear Actuator Mechatronic Structure

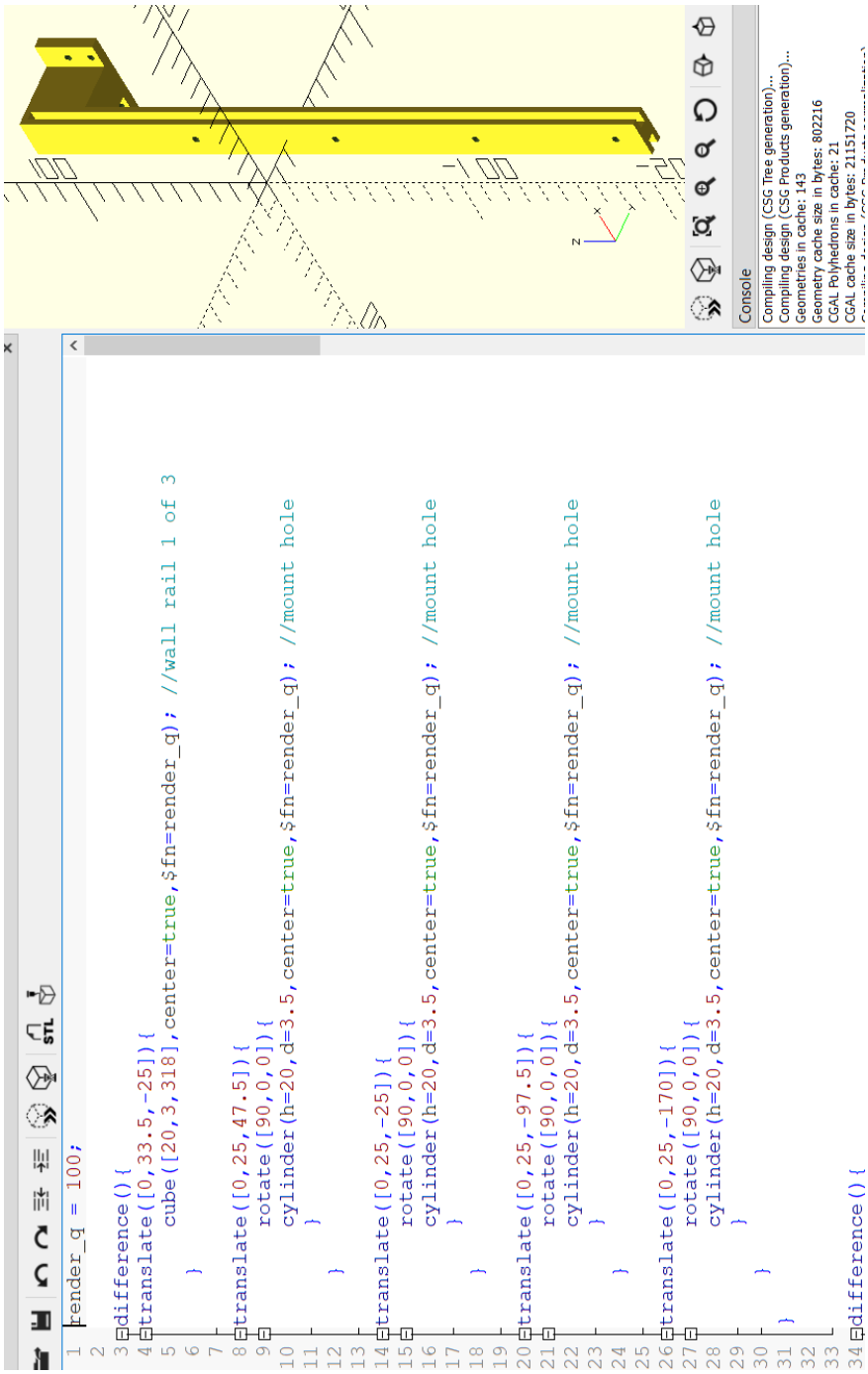


Figure C.6: The rail structure was mounted to the back wall of the ground station so the linear actuators and their brackets could slide up a fixed guided path to open the system lid.

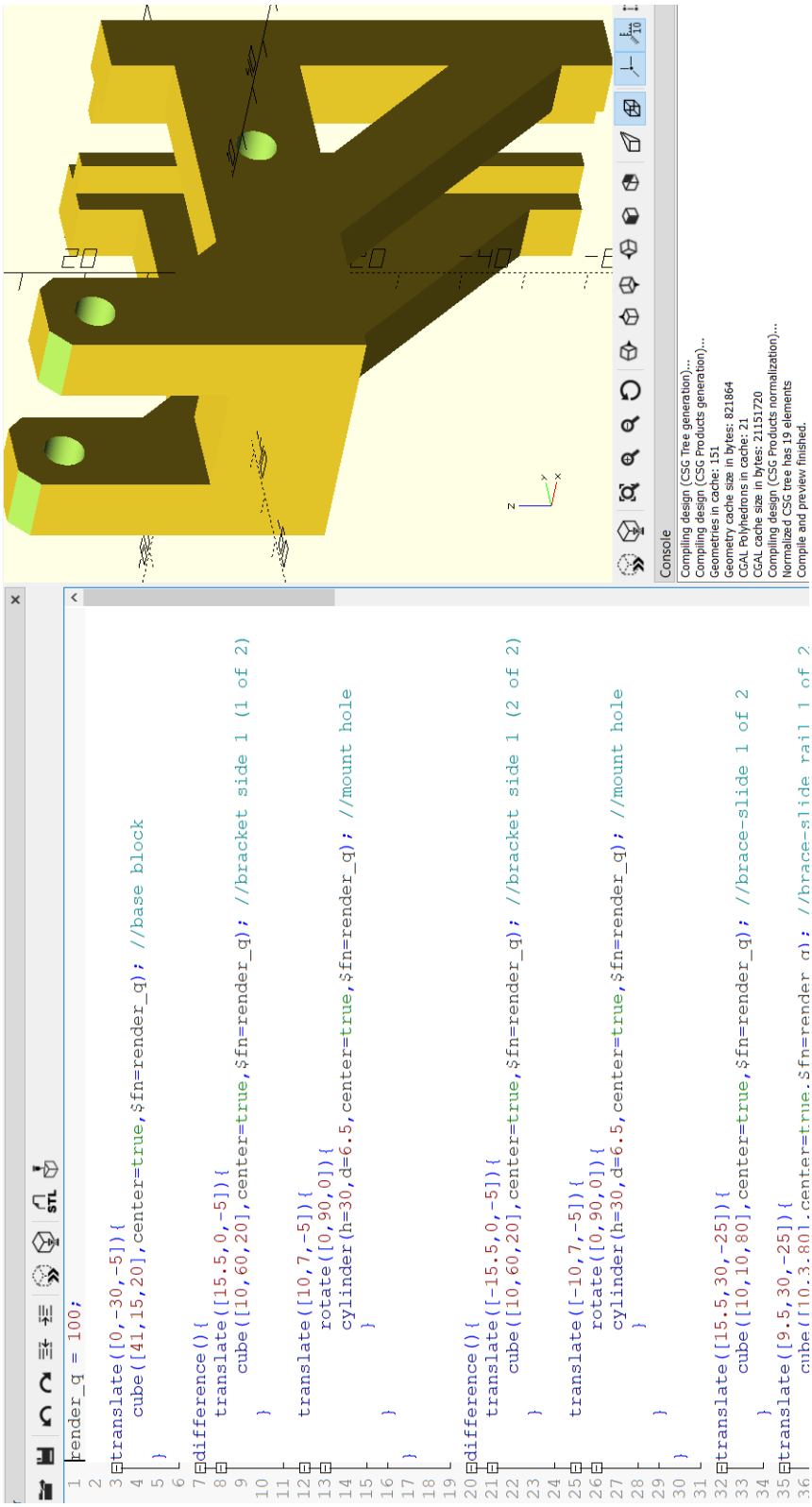


Figure C.7: This is the bracket that acted as a double sided hinge for the bottom and top actuators to pivot from. This structure slid along the rail structure.

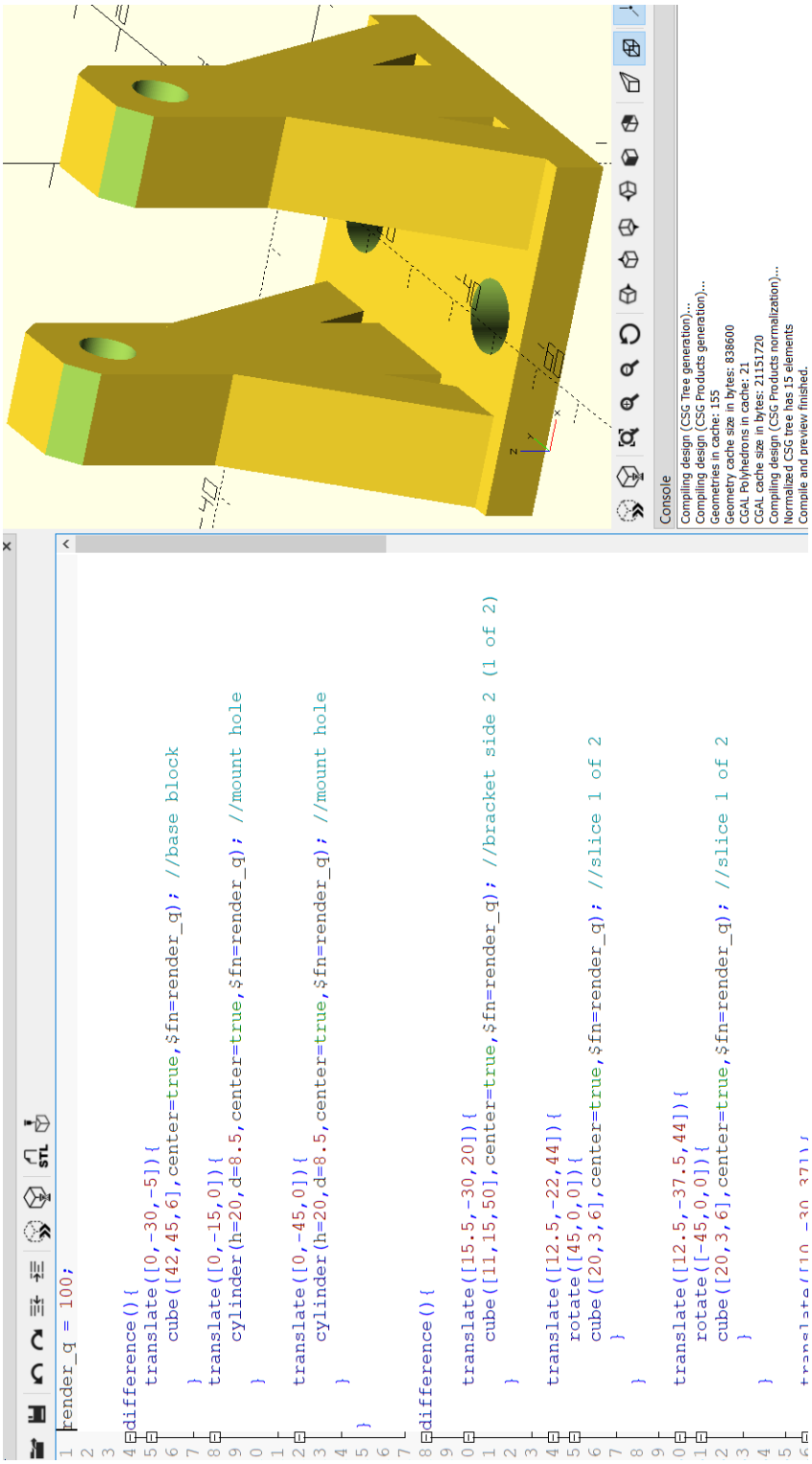
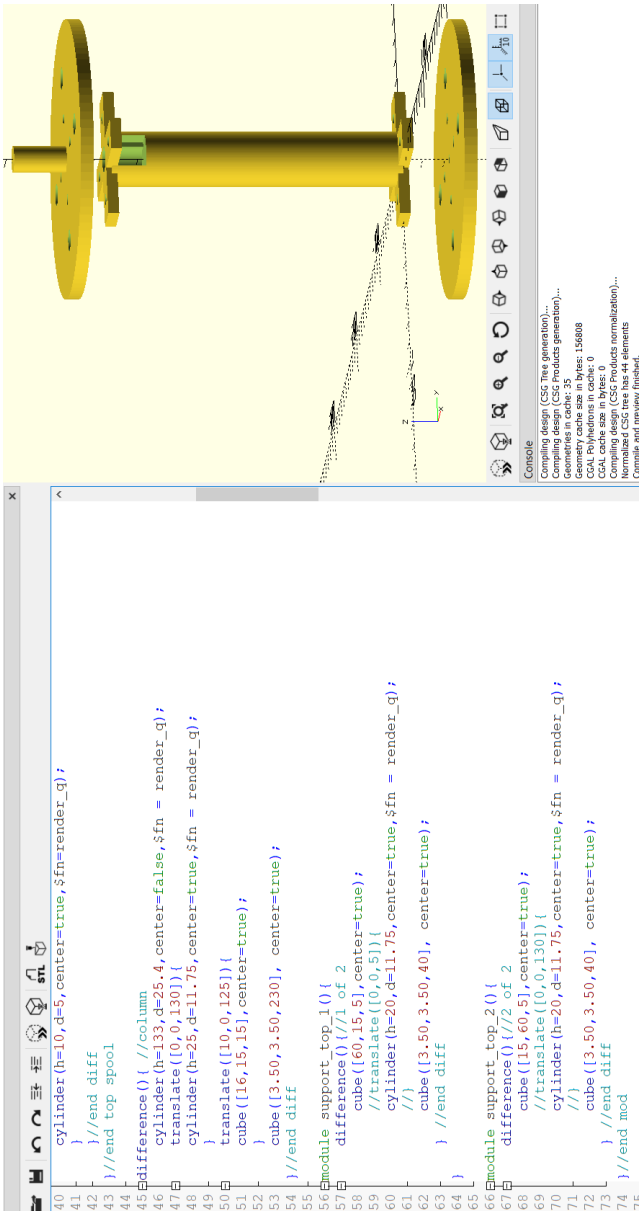


Figure C.8: The top linear actuators mounted to the lid via this hinge bracket where they pivoted.

C.5 Power-Sensing-Tether Spool



```
cylinder(h=10,d=5,center=true,$fn=render_q);
}
}//end diff
}//end top spool

difference(){ //column
    cylinder(h=133,d=25.4,center=false,$fn = render_q);
    translate([0,0,130]){
    cylinder(h=25,d=11.75,center=true,$fn = render_q);
    }
    translate([10,0,125]){
    cube([16,15,15],center=true);
    }
    cube([3.50,3.50,230], center=true);
}//end diff

module support_top_1(){
  difference(){//1 of 2
      cube([60,15,5],center=true);
      //translate([0,0,5]){
      cylinder(h=20,d=11.75,center=true,$fn = render_q);
      //}
      cube([3.50,3.50,40], center=true);
  } //end diff
}

module support_top_2(){
  difference(){//2 of 2
      cube([15,60,5],center=true);
      //translate([0,0,130]){
      cylinder(h=20,d=11.75,center=true,$fn = render_q);
      //}
      cube([3.50,3.50,40], center=true);
  } //end diff
}//end mod
```

Figure C.9: The primary spool was designed as a set of 3 structures which can be seen in the figure. The power-sensing-tether was fed through the cylindrical conduit and through the opening in the column before it was managed around the column. In order to allow for continuous full rotation the tether was transformed through a electromechanical slip-ring before being ran through the structure.

C.6 Onboard Buck Mount

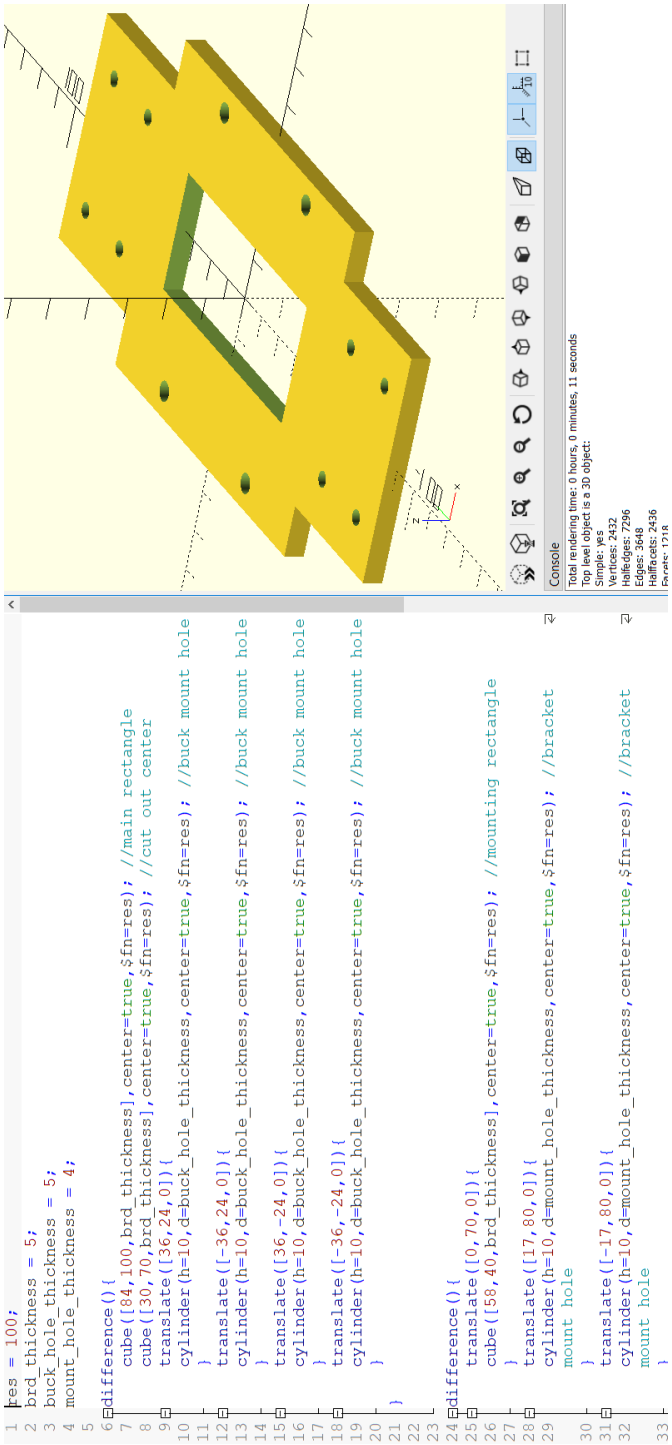
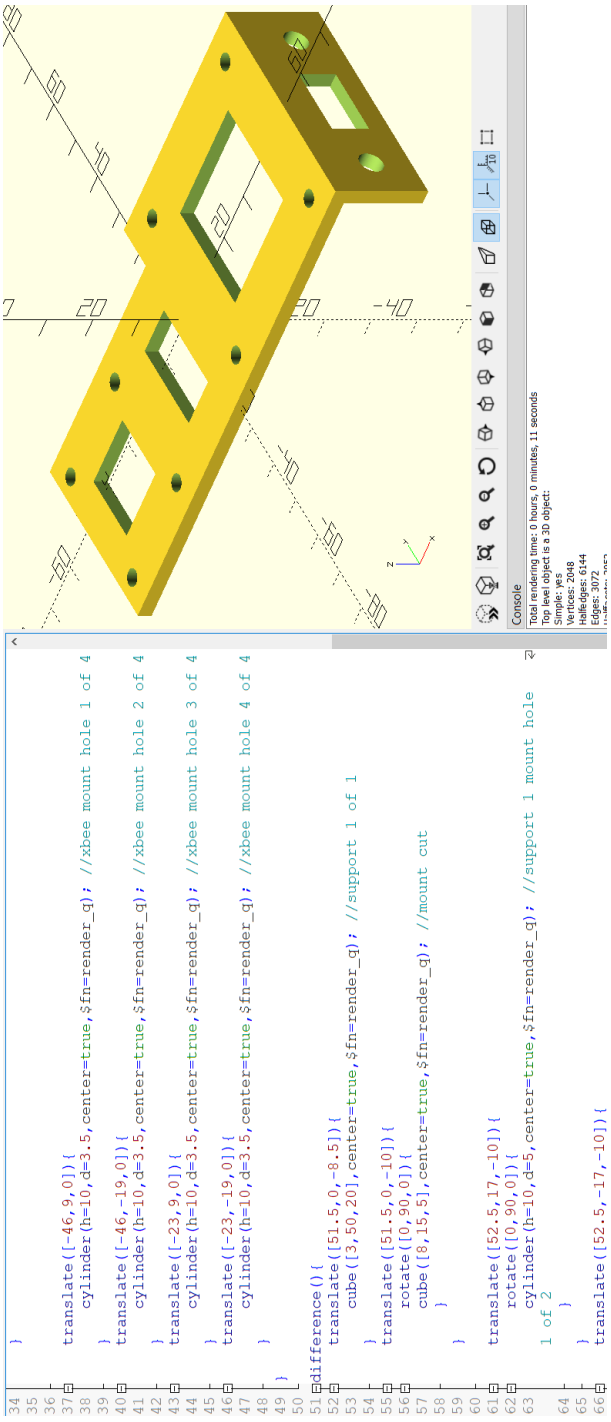


Figure C.10: The onboard Buck step-down power electronics were mounted to the F450 UAS base plate with this structure.

C.7 Onboard RTK-GNSS Mount



```
}
translate([-46,9,0]){
    cylinder(h=10,d=3.5,center=true,$fn=render_q); //xbee mount hole 1 of 4
}
translate([-46,-19,0]){
    cylinder(h=10,d=3.5,center=true,$fn=render_q); //xbee mount hole 2 of 4
}
translate([-23,9,0]){
    cylinder(h=10,d=3.5,center=true,$fn=render_q); //xbee mount hole 3 of 4
}
translate([-23,-19,0]){
    cylinder(h=10,d=3.5,center=true,$fn=render_q); //xbee mount hole 4 of 4
}
}
difference(){
translate([51.5,0,-8.5]){
    cube([3,50,20],center=true,$fn=render_q); //support 1 of 1
}
translate([51.5,0,-10]){
    rotate([0,90,0]){
        cube([8,15,5],center=true,$fn=render_q); //mount cut
    }
}
translate([52.5,17,-10]){
    rotate([0,90,0]){
    cylinder(h=10,d=5,center=true,$fn=render_q); //support 1 mount hole
1 of 2
    }
}
translate([52.5,-17,-10]){
```

Figure C.11: This structure was mounted onboard the F450 UAS and it mounted an Xbee field radio and the Rover-RTK-GNSS module.

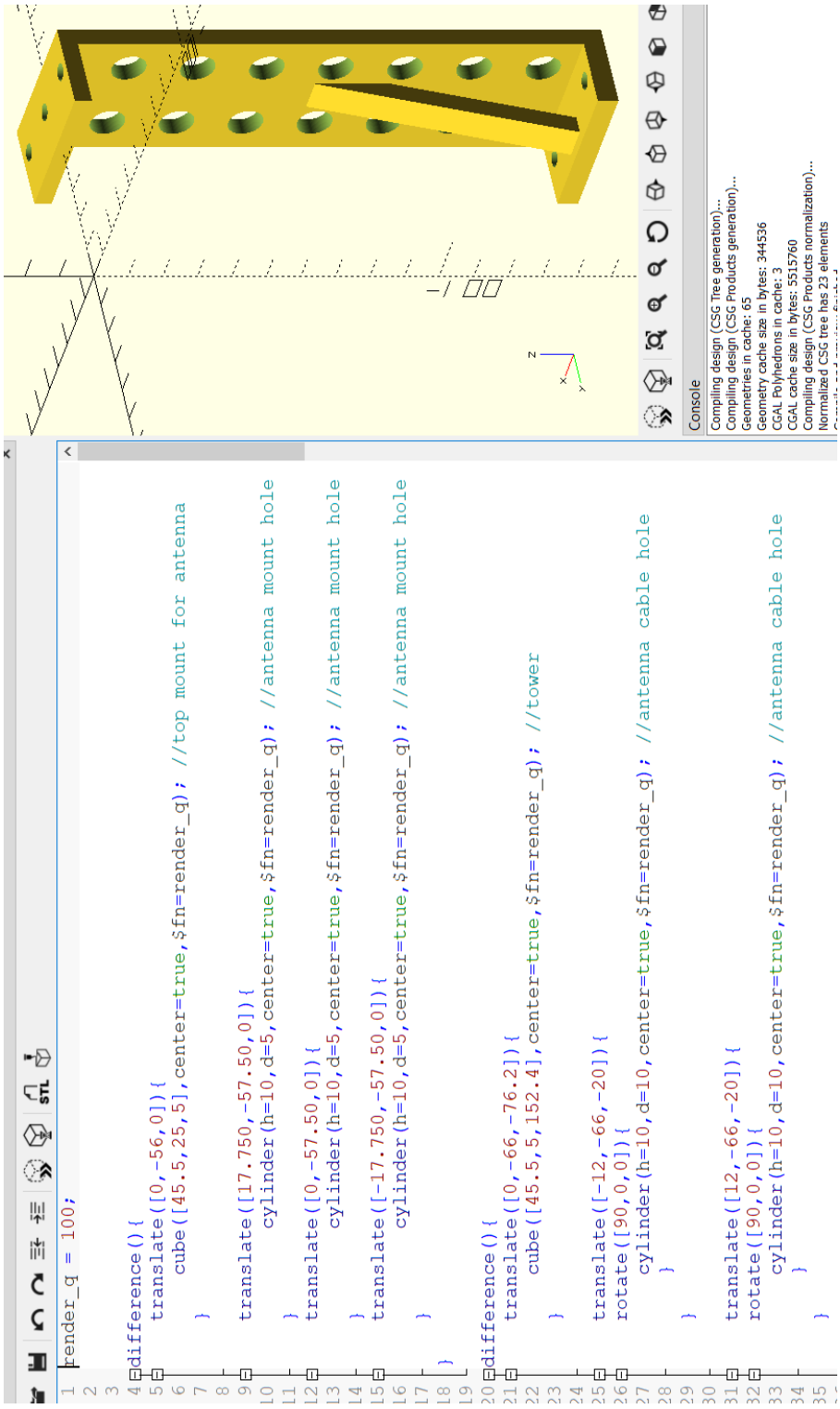


Figure C.12: This structure was developed to manage and distance the sensitive GNSS antenna and SMA-coaxial cable from the power conductors.

C.8 Base Station RTK-GNSS Mount

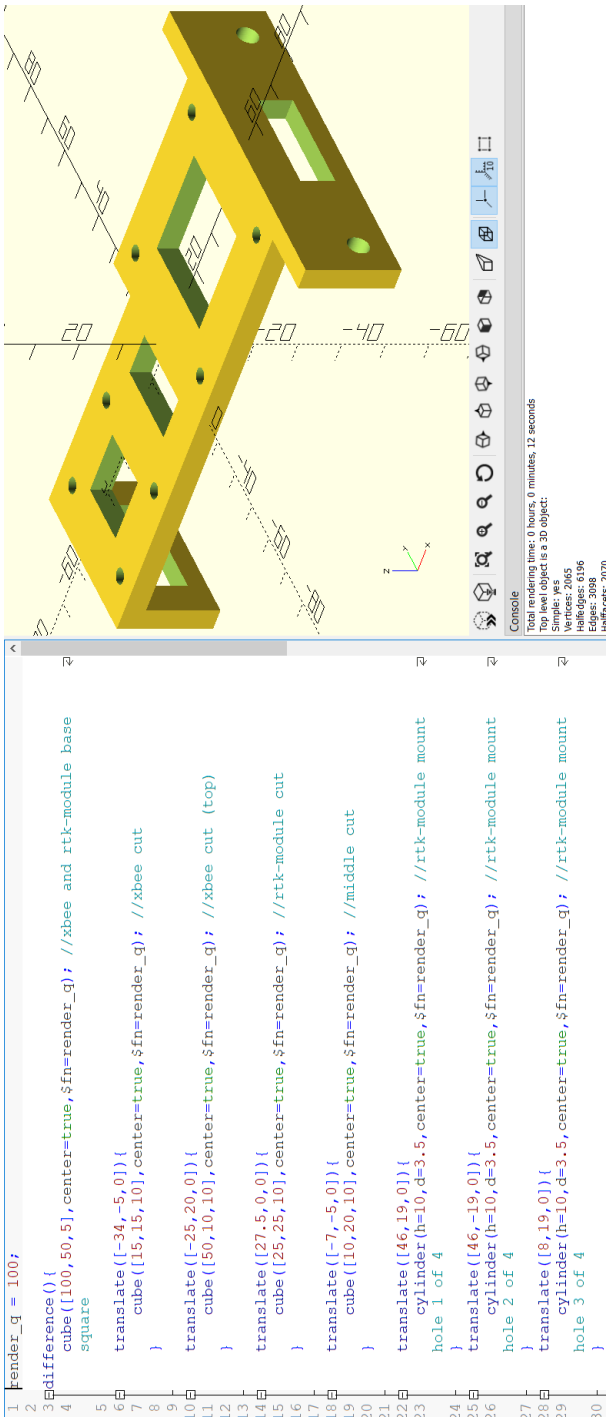


Figure C.13: Similar to the onboard structure, this structure mounts the paired Xbee field radio and base RTK-GNSS module.

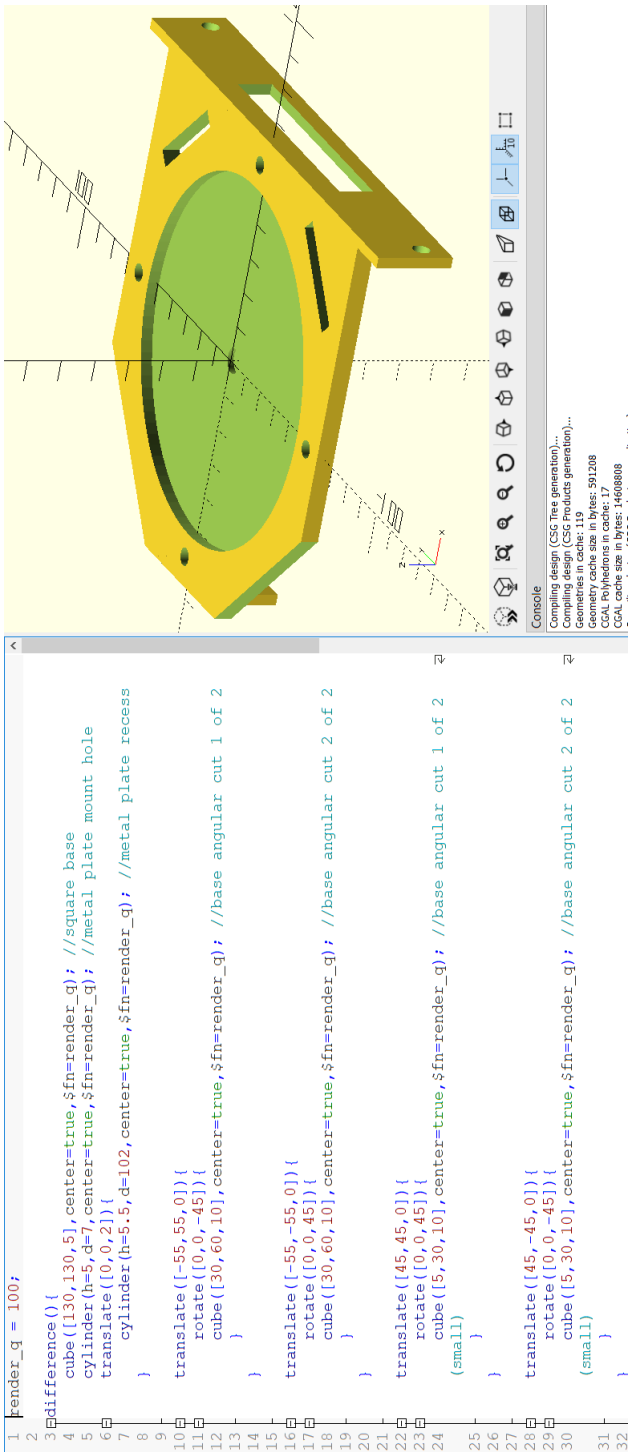


Figure C.14: Since the base station is not onboard the UAS, a metal ground plate can be used to enhance the GNSS antenna reception. This structure was used to mount the ground plate firmly.

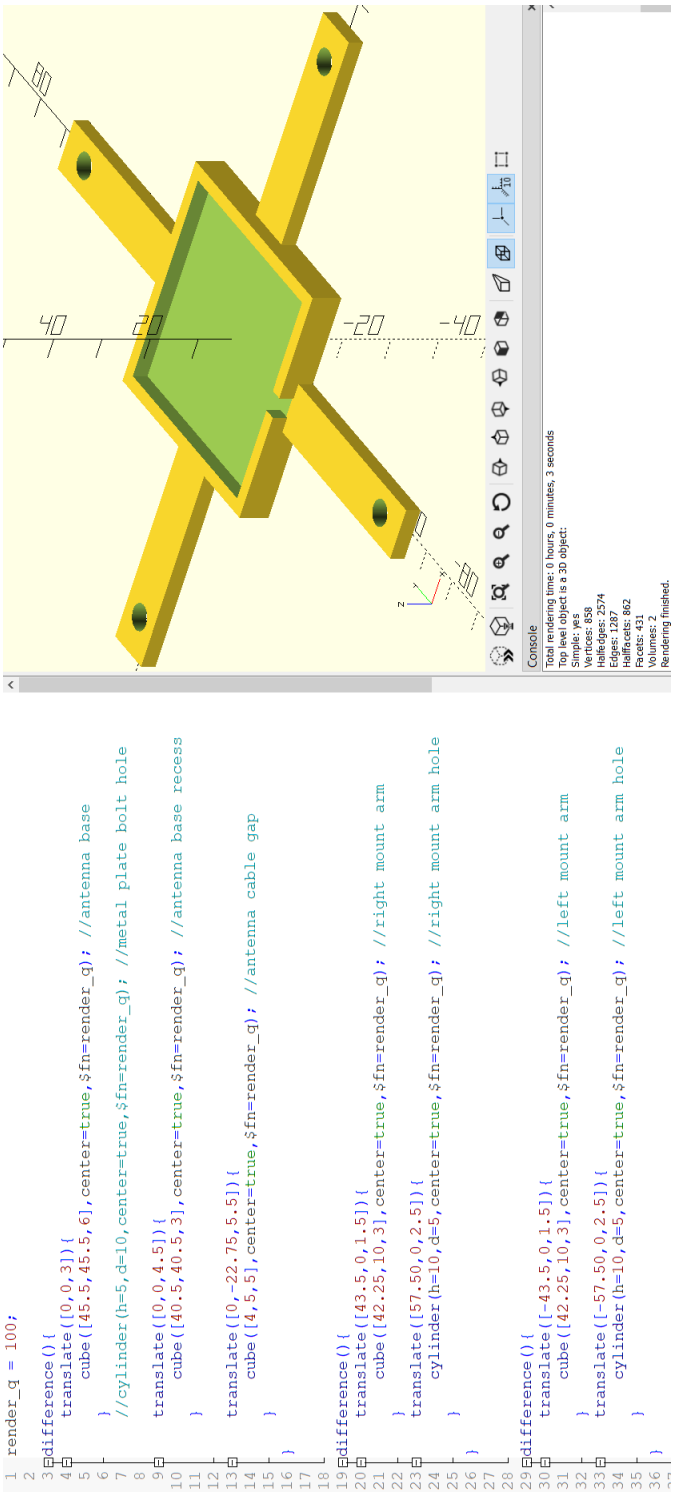


Figure C.15: This structure was used to mount the base station RTK-GNSS antenna to the metal ground plate.

Appendix D

Autobiography

D.1 Background

I was born on August, 12th 1993 in Omaha, Nebraska. I graduated from Omaha Benson High School in May 2012 with the Susan Buffet and Donald F. and Mildred Topp Othmer scholarships to the University of Nebraska - Lincoln (UNL). As an undergraduate I had a strong desire to solve problems, build robots, and understand energy and power systems, so I studied Electrical Engineering. At various points throughout the degree I earned additional scholarships to traveled abroad to Japan, Costa Rica, and Iceland where I gained invaluable global perspective, field research experience, and energy knowledge, respectively. While an undergraduate I also conducted a significant amount of research as a member of the Hydroinformatics and Integrated Hydroclimate (HIH) laboratory under Dr. Francisco Munoz-Arriola and as a Ronald E. McNair Scholar. Towards the end of my degree I spent time working in industry as an Electrical Engineer at Blattner Energy and Omaha Public Power District (OPPD). I graduated from UNL with a B.S. in Electrical Engineering and with minor(s) in Energy Science and International Engineering. Immediately after graduation I transitioned directly into a master's degree in Computer Science & Engineering at UNL funded by the National Science Foundation (NSF) where I

sought to expand my software capabilities while conducting aerial robotic research. I was a GRA under Dr. Carrick Detweiler and Dr. Francisco Munoz-Arriola and a member of the Nebraska Intelligent MoBile Unmanned Systems (NIMBUS) and HIH Laboratories (see Figure D.1). I will be graduating with my masters in Computer Science & Engineering with a specialization in Computer Engineering on Friday, August 13th 2021.

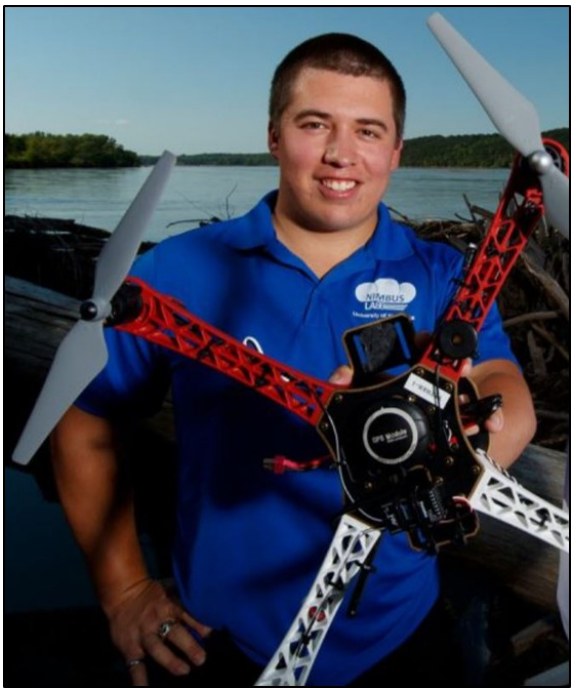


Figure D.1: Daniel Anthony Rico, B.S. 2017, M.S. 2021

D.2 Future Direction

I will start a Ph.D. in Computer Science at UNL in August 2021 and remain a GRA under Dr. Carrick Detweiler and Dr. Francisco Munoz-Arriola and a member of the NIMBUS and HIH Laboratories. After this Ph.D., I will pursue an advanced degree in Nuclear Physics and Engineering so I can contribute to next generation nuclear fission and first generation nuclear fusion reactors. I have and will continue

to dedicate my life to making our world a better place by solving our most critical problems.

D.3 Contact Information

Email: daniel.rico05@gmail.com

Phone: (402) 699-8108

NIMBUS Website: https://nimbus.unl.edu/people/

HIH Website: https://engineering.unl.edu/hih/about-us/

Personal Website: http://www.DanielAnthonyRico.com

YouTube: https://www.youtube.com/channel/UCm9fqWM_iHDmij5INm2lP6w

GitHub: https://github.com/drico7/

Linkedin: https://www.linkedin.com/in/daniel-rico-1b7a1573/