DESIGN AND IMPLEMENTATION OF A 6U CUBESAT FOR LOW EARTH ORBIT COMPUTER VISION

by

Jackson O. Parker

(Under the Direction of Fred R. Beyette Jr.)

ABSTRACT

Small Satellites, especially those at or below the 50kg weight limit for micro-satellite classification, provide low cost alternatives to space exploration and spacecraft innovation. These missions allow programs to achieve higher throughput for repeatability, iteration, and higher permissible risk levels. Whether used as a standalone platform or to facilitate technology demonstrations for future missions, the innovative practice of developing Small Satellite technology is improving the aerospace industry. The University of Georgia Small Satellite Research Lab is currently developing a 6U Earth Observation CubeSat utilizing an off the shelf NVIDIA GPU SoC for onboard computer vision processing. The work laid out in this thesis shows how the mission will utilize this new computational technology on a small satellite platform that increases computational abilities and autonomy at an unprecedented low mission price.

INDEX WORDS: Small Satellite, CubeSat, Computer Vision, Earth Observation, High Performance Computation, GPU Accelerated Software

DESIGN AND IMPLEMENTATION OF A 6U CUBESAT FOR LOW EARTH ORBIT COMPUTER VISION

by

Jackson O. Parker

B.S., University of Georgia, 2018

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2020

DESIGN AND IMPLEMENTATION OF A 6U CUBESAT FOR LOW EARTH ORBIT COMPUTER VISION

by

Jackson O. Parker

Major Professor: Fred Beyette

Committee: David Cotten

WenZhan Song

Electronic Version Approved:

Ron Walcott
Interim Dean of the Graduate School
The University of Georgia
May 2020

ACKNOWLEDGEMENTS

It is imperative to mention that the progress on this mission thus far would have been impossible without the students working in the University of Georgia's Small Satellite Research Laboratory (UGA's SSRL). Over 75 students have contributed to the Multiview Onboard Computational Imager (MOCI), whether that is directly or through their help with the Spectral Ocean Color mission. Funding for this mission was provided by the University Nanosatellite Program (UNP) out of the Air Force Research Lab (AFRL). Their provided expertise and guidance through reviews and mission landmarks have not only benefited the MOCI mission but also the future of UGA's SSRL. Further credit should go to Alex Lin for taking over as Systems Engineer and making sure that the work done for MOCI thus far will lead to successful integration and testing over the next year. Caleb Adams also deserves credit for helping establish the SSRL and the MOCI mission, providing me with a framework to build within.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION AND BACKGROUND

1.1 Description of research problem

Satellites play a variety of vital roles for modern society. Notably acting as the backbone for global connectivity, providing a way to learn more about the cosmos, and acting as a valuable tool to monitor our own planet. The development of large satellites, whatever the purpose, takes extensive resources in time, personnel and funding due to the intricate and robust systems required to function in space for an extended period of time. A recent example of this would be the James Webb telescope being developed at NASA Goddard. This satellite will have the most advanced optics system ever developed on a satellite and has over 100 critical deployment mechanisms that need to be successful for data collection to begin [14]. Billions have been spent on this mission and countless people have been involved from contractors spread across 27 states in the US alone [13]. The reason why we do not see a wide variety of these large budget missions in development is due to the risk involved in funding experimental missions of such expense.

Today companies, including NASA, and universities around the world are navigating around this issue by developing small satellites. Most satellites that fall under this category weigh at or below 500kg with costs typically ranging from a few tens of thousands to 10 million USD, mostly correlated with weight and complexity [1]. With that, the aversion to risk can be lowered in comparison to larger satellites. This allows for quick

turn-around missions focused on incremental/small technology demonstrations with experimental, less risk averse, payloads in a space environment.

Throughout this thesis, the design and implementation of the University of Georgia Small Satellite Research Lab's Multiview Onboard Computation Imager (MOCI) CubeSat, a subcategory of small satellites, will be discussed. The purpose of this satellite is earth observation through visible light computer vision techniques to detect surface objects and test the feasibility of performing Structure from Motion (SfM) from Low Earth orbit to generate digital surface models (DSMs) through the advancement of onboard processing techniques and hardware. Earth observation in general provides a way to analyze how the earth's static and morphological nature affect the human species as well as our effects on it.

Ways of gathering information vary depending on the purpose of the analysis and the resolution of that data gathering tool. For the most part, on-the-ground information gathering techniques are very localized and tend to be on a microscopic scale when considering the scope of complete earth observation. A good example of this would be ecological surveys meant to determine what kind of animals are living in a specific area. Satellites provide a way to do this on a macroscopic scale to supplement methods for geological and environmental analysis performed on the ground. Earth observation payloads are not always visible light imagers but still often generate a lot of data. To downlink the raw data generated by these payloads, high frequency transmitters are required on the satellite, which present their own fiscal and power related costs. Even with the ability to downlink the data, processing is still required to generate something useful, making quick response based on the satellite's data very difficult. Without the

satellite acting in a more autonomous nature the possibility of quick response is very difficult, if not impossible. The MOCI mission is also acting as a technology demonstration for a cheaper way to achieve autonomy and high-performance computation in an earth observing satellite.

1.2 Why is it important?

Even though we, as a nation, have set our long-term goals on Mars as the next potential planetary colony, most humans will be staying on earth for the foreseeable future. Understanding our home can further improve our resource utilization in a way that does not threaten the health of the ecosystem, which becomes increasingly important as the population grows. Having a cheaper and more intelligent way of gathering data can lead to opportunities for more coverage in orbit and more comprehensive data products as well as offering a few more benefits. For instance, if a system like MOCI is proven to work well, tasks that require fast response like disaster alert, forest fire tracking, weather alerts or even surveillance can be made possible using onboard processing. For things that do not require fast response like long term climate analysis, transmissions can be minimized to useful and non-redundant data products. By proving that high-performance computational capabilities can be achieved with cheap and easily interfaced components, doors can be opened for small satellites in a larger scope of applications. Allowing for more computationally complex and autonomous action, onboard high-performance computational units (HPCUs) could be used to enable the deployment of spacecrafts for observing and analyzing other planetary bodies. Since you would not want to transmit many images or complex sensor readings from something orbiting Mars or even one of

Jupiter's moons, data can be refined or minimized to non-redundant and useful information.

Even if the small spacecraft is not meant for earth or planetary body observation, higher levels of autonomy could increase the potential applications of payloads by reducing direct control and allowing for the craft to utilize more sensory information to facilitate action. As can be seen with the various companies that have emerged solely generating small satellite technology as their product, the success of MOCI would act as a technology demonstration for a small, powerful, standalone spacecraft.

1.3 Alternative approach to addressing the problem

MOCI is far from the first aerospace vehicle carrying an earth observation payload. The size of the satellite is primarily dependent on the required peripherals and size of the payload on board. Larger satellite missions tend to allow for more opportunities in larger optical trains or multiple payloads, allowing for increasingly useful data products. A good example of a satellite mission with multiple payloads utilized for earth observation is Terra, launched in December 1999 and beginning data collection two months later.

Table 1 - Terra Satellite Payloads

Terra Payloads	Purpose(s)
ASTER – Advanced Spaceborne Thermal Emission and Reflection Radiometer	High resolution multi-spectral imager mostly between the visible and infrared (Japan)
CERES – Clouds and the Earth's Radiant Energy System	Measure Earth's radiation budget (US)
MISR – Multi-angle Imaging SpectroRadiometer	Imaging earth with 9 cameras off-nadir for aerosol particle investigation, cloud heights, and distribution of land cover (US)
MODIS – Moderate-Resolution Imaging Spectroradiometer	Imaging the entire earth every 1-2 days in 36 discrete spectral bands (US)
MOPITT – Measurements Of Pollution In The Troposphere	Analyzing lower atmosphere using gas correlation spectroscopy (Canada)

Details about these payloads can be seen in table 1. The combination of their readings facilitated important earth observations like effects of climate warming, measurement of Earth's radiation budget, analysis of cloud movement, evaluation of ecosystem health, surface mapping, and much more [28, 43].

A mission like this would be nearly impossible for a university to organize and develop due to required level and variety of expertise as well as funding acquisition. However, the development of a single payload would be a different story. Of course, satellites are not the only usable technology for this purpose as drones have also been often used for both surveillance and local environment analysis from an eagle eye perspective. Drones have been known to carry a variety of payloads for aerial sensing and often generate high resolution data due to their proximity to the ground target. Drones are common platforms for Structure from Motion (SfM) implementations, a pipeline MOCI is utilizing, for a variety of purposes. One specific implementation published in 2017 proved that drone SfM can be utilized as a low-cost method for monitoring greenhouse gas emissions caused by deforestation and forest degradation [30]. The largest downside to drones is power consumption and inability to prolong flight for long periods of time, which is a problem that satellites do not have as one could take years to fall from an ISS-like orbit. Not to mention a vehicle in orbit will be able to cover much more ground as a satellite at that orbit will be traveling over 7 km/s and orbit the earth once about every 90 minutes.

When considering alternative approaches to observing the earth off the ground, the first differences that should be noted are in the sensory payloads themselves. Two primary categories of sensing mechanisms for this purpose are passive and active

sensors. The real difference between the two types is that passive sensing only reads energy that is available in the environment while active sensing generates that energy and reads what is reflected. Cameras and traditional imagers are all performing passive sensing, like MOCI with its traditional visible light camera system. The primary difference between the various types of passive sensors is the section of the electromagnetic spectrum that the sensor is designed to read. The best sensor for the imager really depends on the application. For instance, NIR or IR sensors are great for thermal imaging. A good example of one of these imaging payloads would be the MODIS payload on Terra mentioned above. These can be useful forest fire detection, drought monitoring, and global warming analysis. However, they are not as useful for elevation mapping as surface temperature does not always have a direct correlation with elevation, especially with human habitation. Another interesting type of passive sensing mechanism that is used for application is the concept of multi- and hyper-spectral imaging. This type of sensing captures bands across the electromagnetic spectrum for each pixel, usually outside of the visible light bands. The major difference between multispectral and hyperspectral being the number and width of the bands. A good example of an advanced multispectral imager is Sentinel-2's payload having 13 bands ranging from 443-2190nm and is being used to capture data for monitoring European agriculture and vegetation, landscape changes, water quality, and disaster indicators [45]. The PRISMA mission's payload represents a good example of a hyperspectral imager collecting data containing 249 bands ranging from 400-2505nm and being used for forest disturbance identification, biomass analysis, crop mapping, water quality analysis, climate change analysis and much more [38]. As one can assume, this can lead to an explosion of data and requires

more rigorous data storing, downlinking, and collection techniques. Payloads with these types of imagers are extremely useful for a variety of ecosystem and environment health analysis but have a lot more information than what is necessary for most computer vision tasks. The previously mentioned passive sensing mechanisms have their applications in earth observation. However, their cost is often much higher and visible light imagers provide enough information to accomplish the tasks MOCI is attempting to achieve.

Now considering active sensing mechanisms. This technology can be very effective at a task like DSM generation by emitting radiation and reading what was reflected to derive 3D information. Primary examples of this include light detection and ranging (LIDAR), radio detection and ranging (RADAR) and synthetic aperture radar (SAR). Due to LIDAR using a higher frequency emission than radio waves, it can achieve a higher resolution at short distances, quite useful on drones and robotic platforms. However, as distance from the target increases so does the required power, cost, and size of the transmitter. These have been utilized on missions like NASA's CALIPSO [7] and ICESat [58] missions, but the general concept is unreasonable for application on a small satellite platform. As RADAR operates using radio waves rather than the near-visible light waves LIDAR emits, they are more equipped for to observe targets at larger distances but often have lower resolution readings. SAR is the primary type of RADAR system being developed and utilized in satellite technology today. By taking RADAR reading snapshots as it passes over an area a larger aperture is "synthesized", allowing for higher resolution data collection. Unlike LIDAR, SAR has seen implementations designed for small satellites like the X-Band SAR for the 100kg class [37]. One of MOCI's

goals is to prove a much cheaper way of achieving exactly what this technology has been purposed for on satellite platforms, by utilizing passive visible light imagers.

1.4 Brief overview of the approach undertaken in this research

1.4.1 Background on image processing needs

When considering computer vision or image processing from satellite imagery, one thing is important to realize, there is information in every pixel. Due to this, larger camera sensors are ideal to allow for more information to be derived from each image. As with any other sensor, noise is a problem to deal with, requiring the existence of filtering methods. Removing all noise is a difficult issue when considering images, due to that noise just being subtle changes in brightness of individual pixels. To mitigate this, accurate and robust feature detection algorithms are required.

Processing high resolution imagery requires an incredible amount of repetitive operations on different pieces of similar information. Some image processing tasks, like dense feature detection or matching, take an unreasonable amount of time on a typical CPU. Due to this, the utilization of a hardware accelerator to provide a SIMD processing model is especially useful, allowing simultaneous execution of the same instruction on multiple pieces of data. This significantly reduces the time that execution takes for repetitive tasks, like those often found in computer vision algorithms. MOCI will be utilizing a GPU platform for this purpose.

1.4.2 Background on the use of Small Satellites as a platform for this research

One of the best places to get a firsthand look at the growth in focus on small satellite technology is the Small Satellite Conference, hosted at Utah State University at the end of every summer. Last year was the 33rd year this conference was held, with

many universities attending and over 200 companies involved in the industry exhibiting. Student competitions, a wide variety of expert speakers and countless exhibits provide a fantastic yearly opportunity to see the state-of-the-art technology in this field. Having gone the past two years, I was able to witness how shockingly large the industry is.

With the large number of companies exhibiting there, not many are developing full satellites. Rather they are just focusing on selling specific subsystems to universities and companies focused on developing spacecraft. For the most part these subsystems are what would be considered off the shelf components or COTS parts, in which the interfacing and any customization around the hardware is the user's job.

As many companies are developing small satellite COTS technology, form factor and interface standards have emerged to make designing and integrating a small satellite bus with parts coming from different companies trivial. These trends tend to depend on the specific type of small satellite as there are five primary types: small, micro-, nano-, pico- and femto-satellites. Generalized weight comparisons of these satellites are included in table 2. The specific type that is being focused on throughout this thesis is the CubeSat. CubeSats are defined in size by a U, or a 10x10x11cm cube. The specific satellite being discussed in this paper is a 6U CubeSat, so it teeters between the microsatellite and nanosatellite definition as it will weight around 11 kg. CubeSats are launched in configurations from 1U all the way up to 27U depending on the size of the payload and the subsystems required to support it.

Table 2 - Satellite Classifications

Category	Wet Mass Range
Large Satellite	> 1000kg
Medium Satellite	500-1000kg
Small Satellite	< 500kg
Micro-	10-50kg
Nano-	1-10kg
Pico-	0.1-1kg
Nano-	<100g

In the past 20 years the number of launches for small satellites per year have been increasing. With the throughput that the small satellite development process enables, failures are guaranteed. In a paper published by the conference, University CubeSat Project Management for Success, the authors went over some trends in university launched small satellites. As of 2018, 428 university-class satellites have been launched, with the vast majority being CubeSats. Launches per year are increasing due to many universities beginning to develop programs and launch their first mission. In the 2018 benchmark, 192 universities took part in the total university-class launches with 106 of those only having launched their first mission. Mission success rates have gone up in recent years, but according to this paper, first mission success rate is at about 33% [4].

Both NASA and the Airforce have taken part in encouraging universities to take part in small satellite technology development through programs like the NASA Undergraduate Student Instrument Project (USIP) and the Air Force Research Lab's (AFRL) University Nanosatellite Program (UNP). USIP is described as an Educational Flight Opportunity (EFO), encouraging universities to develop payload technology for flight on sounding rockets, balloons, small satellites, etc. This program most recently chose 47 teams of undergraduate students for these opportunities and provided over $8 million USD in funding. Thirteen of these missions are for the development and launch of

CubeSats, one being a University of Georgia mission. UNP has a similar, but more focused, goal of enabling university programs to develop small satellite technology. This year UNP celebrated its 20-year anniversary. In that time 38 universities, 7 launches and 15 nanosatellites have been enabled by the program through 9 complete flight selection processes and the tenth occurring this year. This program's rigorous process has been refined over the years into an optimal machine, guiding and educating college students to help improve their small satellite develop programs and subsequent missions. This is distinctly shown here at the University of Georgia through the mission being discussed in this thesis. Even though these programs are localized to United States university participants, universities all around the globe are developing small satellite technology. In fact, most university missions are funded privately or through accepted grant proposals from foundations like the National Science Foundation (NSF).

The development of small satellites has by no means been localized to development programs at the university level. Companies like TYVAK [47], Planet Labs [41] and Lynk [26] are developing multiple small satellites per year and launching into orbit. Planet Labs and Lynk are developing constellations of small satellites purposed for enhanced coverage, the former for imaging the entire globe at 3-5m resolution once a day and the latter attempting to provide maximum cell connectivity. TYVAK has a wide variety of designs, some being utilized for constellations and others being purposed for singular platforms. Government organizations like NASA and the AFRL also have small satellite portfolios where a wide range of missions are in development and launching to both test new spacecraft technology and develop cheaper and useful platforms for earth observation. A list of some publicly detailed small satellite designs/missions from the

mentioned companies, organizations and universities can be seen in table 28 located in the Appendix. Like MOCI, many of the listed missions are purposed for earth observations. When considering missions that relate to MOCI's attempt to further onboard high-performance computation with the use of a GPU platform for computer vision, no missions are publicly available, although some GPUs have flown.

Since the creation of the University of Georgia's Small Satellite Research Lab, it has been developing two missions; the first being the Spectral Ocean Color (SPOC) mission while the second, MOCI, is the one primarily discussed in this thesis. Both missions are focusing on intelligent earth observation and solutions within that problem space on small satellite platforms. SPOC is funded through NASA's USIP program and a partnership with NASA Ames. Its primary contribution to the field is the utilization and design of a custom adjustable multispectral imager. The major difference between SPOC's payload and a traditional hyperspectral imager is that it does not record the entire spectral range for each pixel but focuses on a range of spectral bands between 400 and 850 nm at a spectral resolution of 4nm and spatial resolution of 130m. This multi-spectral imager will collect data that can be used to quantify the ecosystem health of coastal regions, specifically looking for algal activity, sediment suspension and vegetation health [9]. SPOC has finalized its preflight preparations and is currently awaiting launch.

MOCI is funded and reviewed through the AFRL's UNP program after winning the NS-9 small satellite flight selection review competition in late 2017. As the second mission, MOCI has gained an incredible amount of knowledge related to the process of integration and verification that will prove decisive to the success of the mission once integration starts sometime late spring 2020 or early summer 2020. Both missions have

overlap in certain subsystems and flight software that have made the primarily parallel development possible.

Primary work behind the design of the MOCI satellite as well as design alterations that have occurred during the review and development process are to ensure that the MOCI mission accomplishes everything mentioned in the following mission statement:

> ***The Multi-view Onboard Computation Imager (MOCI) mission will acquire imagery of the Earth's surface from LEO and perform real time Structure from Motion (SfM) at a landscape scale using custom algorithms and off the shelf, high performance computational units. The MOCI mission will also identify various objects on the earth's surface while training students in STEM related fields. Efficient data compression, feature detection, feature matching, and SfM processing techniques of space-based imagery will be performed on board the spacecraft.***

Table 3 - Mission Success Categorization

Full Mission Success	All full mission success criteria achieved.
Partial Full Mission Success	All minimum mission success criteria achieved, with at least one full mission success criteria achieved.
Minimum Mission Success	All minimum mission success criteria achieved.
Partial Mission Success	One or more mission objectives saw no success.
Mission Failure	No mission objectives saw any success.

The way that the success of the mission is confirmed is through the evaluation of MOCI completing its mission objectives as detailed in table 4. Each objective in turn has its own success criteria for both minimum and full mission success, seen in table 4. As meeting one success criteria does not guarantee that the others are met, there are five different possible outcomes of the mission, as can be seen in table 3. Of course, the worst of the outcomes would be mission failure and could be caused reasons by a variety of problems during launch, but by removing single points of failure and emphasizing precaution during

the development process the goal is to leave the only possibility for this as a launch related failure.

Table 4 - MOCI Mission Objectives

Objective	Minimum Mission Success	Full Mission Success
MOCI shall run on-board SfM of target areas to make 3D terrain models and transmit this processed data product.	MOCI shall generate a Digital Surface Model (DSM) within 1 sigma of existing DSM's	MOCI shall generate a DSM, demonstrated on orbit, within 1.281 sigma (80%) of existing DSM's
MOCI shall identify well known objects using neural networks.	MOCI shall identify an object on the Earth' s surface with 60% predictive accuracy.	MOCI shall identify an object on the Earth' s surface with 90% predictive accuracy.
MOCI shall image the coastal regions of the Eastern United States for marsh phenology, land use, and off-coast water quality.	MOCI shall acquire 3 images, of one coastal target.	MOCI shall acquire 15 images of one coastal target or the Sapelo Island test area over the lifetime of the mission.
MOCI shall train students in STEM related fields by having them investigate optimal data transmission techniques, georeferenced imagery for mapping, conduct photogrammetric processing of images acquired from the satellite, develop community outreach programs, and learn general aerospace manufacturing/testing/designing skills.	30 students shall be directly involved in MOCI satellite development and integration for at least 2 \semesters and the duration of their time at UGA or over the lifetime of the project.	50 students shall be directly involved in MOCI satellite development and integration for at least 2 semesters and the duration of their time at UGA or over the lifetime of the project.
	The MOCI project shall give 5 community outreach presentations, mentor 2 local high school students, and 5 space news/educational podcasts.	The MOCI project shall give 20 community presentations, mentor 5 local high school students, host 2 workshops, release 10 satellite related instructional YouTube videos, and 20 space news/educational podcasts.

CHAPTER 2: SMALL SATELLITE DESIGN

2.1 UGA SSRL

The University of Georgia Small Satellite Research Laboratory (UGA SSRL), was founded in 2016, as a way for undergraduates to take part in spacecraft technology development. In its first stages, one satellite was being planned for geospatial analysis. Then both the SPOC and MOCI proposals were accepted, leading to two missions with different purposes in earth observation. Through the development of the MOCI mission, from concept to current state of pre-integration, more than 75 individuals have been a part of the SSRL and contributed in some way. As any satellite cannot be completely designed or built by one person, it is necessary to mention those who contributed towards the progress and the mission's current state. Throughout this chapter, acknowledgments will be given directly to those majorly involved or leading development on each portion of the satellite. To better understand the contributions made to this mission, understanding the structure of the lab is helpful. Figure 1, shows team structure including the four teams that illustrate the divergent responsibilities including: mission operations (MOPS) team, mechanical team, electronics team and hardware team. The major responsibilities for

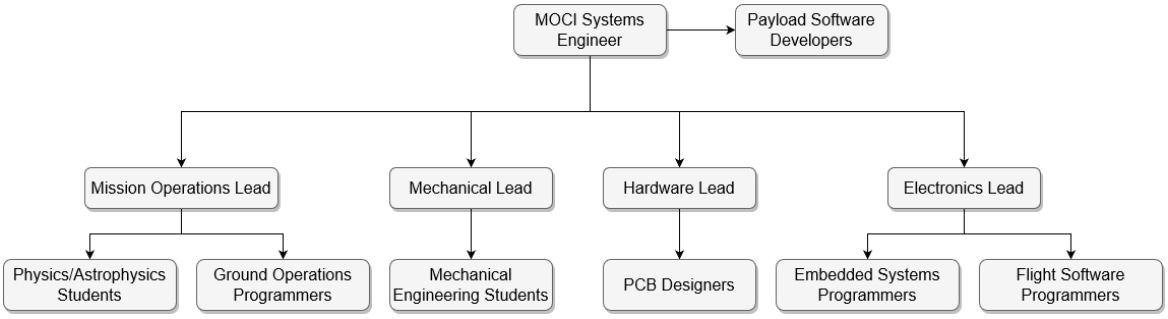

Figure 1 - SSRL MOCI Team Structure

Table 5 - MOCI Team Responsibilities

Mechanical Team	structures, thermals, fabrication and assembly
Hardware Team	interface board design
Electronics Team	flight and embedded software design
Mission Operation Team	licensing, mission planning and ground station preparation
Payload Software Team	experiment research software

each team can be seen in table 5. The leads for each team, appointed based on the expertise and drive they displayed as general lab members, are responsible for ensuring that tasks assigned by the systems engineer, my position, are accomplished in a timely manner. With MOCI's team structure, the systems engineer is responsible for guiding interdisciplinary collaboration, ensuring that inconsistencies are prevented in overlapping design areas, guiding reviews, overseeing testing and ensuring that the design is progressing towards a functional integrated satellite. At the beginning of the sections within this chapter, team leads and any individuals that significantly contributed to that part of the satellite design will be mentioned.

2.2 Defining Design Requirements

Due to the failure rate trend for university programs working on their first mission, and the fact that we are developing our first two mostly in parallel, strong systems engineering processes had to be defined and utilized. This comes in two primary forms, one being weekly reviews of team and interdisciplinary collaborative progress and the other being an actual comprehensive review of progress and design changes two or three times a year. As each team has their own meeting every week, so does leadership with the systems engineer leading that meeting. Even though the systems engineer is heavily present in the lab during the week for stand ups and individual conversations, it is

important that these meetings occur in order to bring up topics in a setting where all team leads are present to hear how other teams are progressing as well as ensure that there is no disconnect when it comes to tasks that are interdisciplinary. For instance, the solar panel design process requires both mechanical and hardware team to be working together to ensure that the panels are functional and that they cover the satellite properly. When there is a disconnect in tasks designs can diverge and serious problems can arise, primarily leading to wasted money and resources due to required redesign. Aside from the weekly meetings that help me, as systems engineer, keep track of where each team is and ensure that tasks are progressing well, the periodic and exhaustive reviews allow for the satellite design to get put under a microscope to expose any deeper and often smaller issues.

2.2.1 UNP Program

One of the major benefits of working within a program like UNP is their proven ability to provide direction for system refinement and reviews. Each university accepted into the program goes through four phases. The flow of those phases and associated reviews can be seen in figure 2. Throughout reviews there are a set of document deliverables that help establish progress and thoroughly illustrate the mission design

Table 6 - The Five Tests

Day in the Life	Test deployment, mode transitions, anomalous behavior handling, and usage of a realistic 24-hour timeline.
Simulated Communications	Testing the functionality of all communication subsystems and their ability to communicate with the SSRL ground station.
Command Execution	Testing all commands that can be sent form the ground station to the satellite.
Complete Charge Cycle	Proving ability of the solar panels to charge the batteries as well as testing full charge and discharge cycles on a simulated solar panel array through a power supply.
ADCS (NS10)	Testing the functionality of the ADCS unit and its actuators.

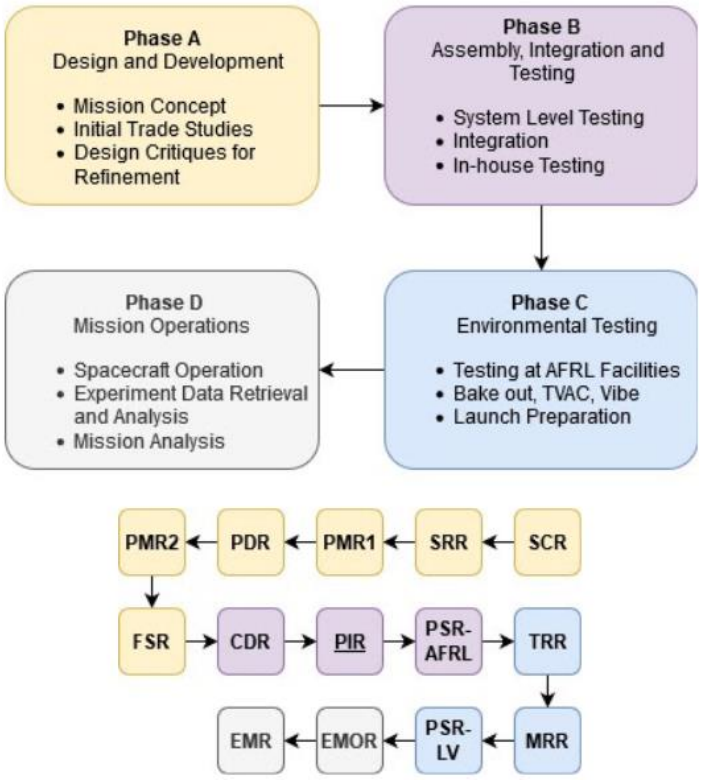

Figure 2 - UNP Phase and Review Flow

concept. Phase A is primarily mission conceptualization and proving the validity of the mission for funding, ending with the flight selection review (FSR) where only a few universities are chosen. Phase B then begins with CDR which illustrates design choices, allowing experts invited from UNP and the AFRL to critique the mission. Once CDR is completed, engineering unit hardware is purchased leading to where MOCI currently stands, pre-PIR. Our next two reviews, PIR and PSR, are both focused heavily on testing and are considered pass-fail reviews. These reviews prove the functionality of the satellite design by executing the five tests outlined by UNP's program, which can be seen in table 6 [6]. The only real difference between PIR and PSR is that PIR performs these tests on the engineering unit while PSR performs these tests on the flight unit. When PIR is passed, MOCI's integration process starts, leading to PSR once fully integrated. To

ensure success during UNP reviews, the SSRL conducts internal reviews every few months.

2.2.2 Requirements Verification

During a large-scale design process, it is important to set requirements early. The requirements help add constraints to the many design processes to ensure that the final product functions to facilitate the success of the mission and completion of mission objectives. At this point, the most important deliverable involved in our review process is the requirements verification matrix (RVM). This document is comprised of a hierarchy of requirements, starting at the mission objective level, stepping down towards system requirements and ending with a series of specific quantitative subsystem requirements. By referencing this document, one can find a comprehensive list of quantifiable design requirements for each subsystem necessary to make MOCI function as it needs to. For the purpose of this thesis and to provide a more generalized view of what is required to accomplish each mission objective, a subset of all the requirements are shown in table 7.

Table 7 - High Level Requirements For MOCI Mission Objectives

For all objectives	• MOCI shall be able to power and charge itself. • MOCI shall be able to communicate with UGA's ground station. • MOCI shall survive launch and successfully burn up during deorbit within 25 years.
MO-1 **Orbital Structure from Motion**	• MOCI shall be able to take grayscale images. • MOCI shall be able to track a target for a sequence of images to be overlapping.
	• MOCI shall be able to utilize the onboard GPU SoC.
MO-2 **Surface Object Identification**	• MOCI shall be capable of RGB images. • MOCI shall be capable of keeping its orientation normal to the ground.
MO-3 **Image Coastal Regions**	

Throughout the rest of this chapter, these requirements will be discussed and used to justify decisions made on the design of MOCI.

2.2.3 Orbit and Operating Environment

With MOCI being a mission funded by UNP, launch provider and launch scheduling are handled by the AFRL after handoff. All that UNP requires from us in order to launch is a range of orbital parameters within LEO. The orbit target for this mission is one with an apogee and perigee of 400-525km and an inclination of 50 +/- 15 degrees. This will place the satellite in low earth orbit (LEO) very similar to the International Space Station (ISS). At that altitude, MOCI will be orbiting at just about 7.67 km/s and have a period of about 93 minutes. In addition to the geometry and orbital localization, it is also important

to consider the harsh environment factors inherent to an orbit at that altitude, most notably radiation and lack of convective heat transfer.

High energy particles from galactic cosmic rays, solar wind and those trapped within the Van Allen belts represent a significant consideration for any spacecraft. Due to MOCI's targeted orbit the South Atlantic Anomaly (SAA) is the primary concern for high energy particle effects on electronics, passing through this multiple times a day [27]. Strengthening the design based on this inevitability is a necessity due the damage that a high energy particle can do when it collides with an electronic component. This threat usually comes in the form of a single event effect (SEE), which is a generalization term that encompasses single even upsets (SEUs), single event latchups (SELs), and single event interrupts (SEI). These all can have data corrupting effects on memory components, or even render a processor non-functional [50]. Due to this, electronic components must be chosen that are at least radiation tolerant. Meaning that the components must be functional in an environment up to a certain level of radiation, if not any radiation level. Radiation hard components tend to be extremely expensive and should only be used if necessary and to remove single points of failure. Radiation shields are also used to mitigate this problem and protect components that are neither radiation tolerant nor hard but are required for the functionality of the satellite.

Due to the lack of a significant atmosphere, or fluid to allow convective heat transfer, the only methods of heat transfer for a satellite include internal conduction and radiation. To keep all components of a satellite within their operating and storage temperature bounds, the usage of passive and/or active thermal control systems are necessary. When considering the upper limit of this problem, the components that

consume more power also generate more heat. At the lowest level, the primary contributor to this is the concept of Joule heating, which generally can be considered heat caused by friction between electrons and the atoms of the conductor which they are passing through [53]. Passive cooling systems usually rely on increasing opportunity for thermal dissipation through conduction utilizing tools like heat straps, heat sinks, thermal paste, etc. to encourage the dumping of heat to the frame. Active cooling systems like cryogenic coolers contact components that generate a lot of heat to encourage more than what passive cooling can achieve. When regarding the lower limit of this problem and ensuring that components are kept above a certain temperature, active methods are necessary. This is especially necessary when the satellite is in the earth's shadow (eclipse) and does not have the sun's radiative energy for heating. Most often batteries require heating as they have the highest lower end for storage and operating temperature, so many space rated batteries include heaters on the units. In eclipse ambient temperature can get as low as -11 degrees Celsius, while in the sun can be as high as 30 degrees Celsius [51].

2.2.4 Launch & Deorbit

It is of course also important to consider the process of getting to orbit. As a CubeSat, MOCI has several options for launch vehicles and deployment mechanisms. Although UNP takes care of finding MOCI's "ride", understanding constraints with typical deployers and launch providers help to ensure that launch once flight ready occurs in a reasonable amount of time. Due to the cost and typical size of modern launch vehicles, it unreasonable and wasteful to develop one for a single small satellite mission. Ride-sharing provides a cost efficient way for small satellites to get to orbit by sharing the space

in the launch vehicle with other payloads. The primary customer of the launch vehicle could be that of a large spacecraft developer or a company focused on dedicated ride-sharing launches for smaller spacecraft. The former has been more common, but according to NASA 62% of the 466 satellites launched in 2017 were categorized as nanosatellites, so it is expected that more launch providers will be dedicated to small satellite launches in the future [43]. One major difficulty with small satellite launches is the ability to target a specific orbit, which is why MOCI has window of acceptable orbital parameters to UNP. Although unnecessary for MOCI, solutions to this problem exist in the form of orbital maneuvering systems like those being developed by MOOG [33], or small satellite propulsion systems like cold gas, resistojets, electrospray, and vacuum arc thrusters that have been utilized by CubeSats [46].

Companies that offer launch integration services allow a level of abstraction from considerations of the actual launch vehicle for small satellite mission developers. By fulfilling the requirements set by these companies, we can make sure that a mission like MOCI will easily find a ride, survive launch and successfully deploy. Deployers that release CubeSats are often in the form of canisters, with a door on one side and springs loaded internally to eject the satellite. Although deployers with a 1Ux1U insertion form factor are ruled out for MOCI due to its 6U configuration, many dispensers do support this common size. The rails of a CubeSat are often the portions of the satellite that make contact with the deployer, on all four corners along the Z axis. In other cases, a tab system can be utilized, but MOCI is using the rail configuration. Deployer compatibility specifications and standards usually revolve around these as they act as the primary load points and contacts. For instance, NanoRacks, the launch integration service that the

SSRL's SPOC mission, has requirements for 3U DoubleWide (6U) satellites with specifics surrounding the rails, electronic interfaces, and other safety precautions. Some important requirements can be seen in table 8 found in their interface definition document [34]. Specifications like these and those advised by UNP, like a structural 100Hz fundamental frequency minimum and a minimum load allowance of 30g, allow for proper requirements to be derived.

One of the reasons that MOCI has set an orbit restriction of 525km or below is due to the "25-year rule" set by NASA. This rule simply states that once a mission is terminated, or its experiment has completed, it must deorbit within 25 years, or 30 years after launch if unable to reach a graveyard orbit [35,36]. If MOCI was to be deployed above 525km, a deorbit mechanism would be required. Most of these deorbit mechanisms utilized by small satellites are passive and add surface area to the satellite in order to increase collisions with the sparse atmospheric particles and increase drag. For instance, the NASA Exo-Brake or the UTIAS-SFL Drag Sail [19]. An active solution would be a propulsion system, which would be a costly addition, both fiscally and in terms of space, to a small satellite just for deorbit. MOCI is currently relying on that lower orbit and its natural orbital decay from the drag associated with its form factor. This is a very

Table 8 - NanoRacks 3U DoubleWide in Rail Configuration Deployer Requirements

Structural, Mechanical and Electrical Systems Interface Requirements				
Rail Length (Z dimension)	366mm (+0/-65.0)			
Rail Dimensions (X and Y)	8.5mmx8.5mm (+/-0.1mm)			
Rail Edge Radius	0.5mm +/- 0.1mm			
Z face Rail Ends	(+Z only) Completely Bare w/ Area of 6mmx6mm Coplanar with the rail ends within +/- 0.25mm			
Roller Switches and Continuity	Rails shall be continuous other than rail mounted deployment switches which must not impede the smooth motion of the rails across the deployer guard rails contacts.			
Z Axis Extension	2mm off the Z faces with exception to the load points on the +/- Z face of the payload			
Contacts with Deployer	Rails shall be only contact in the X and Y axes, other than deployment switches if used			
Rail Hardness	Greater than or equal to hard-anodized aluminum (Rockwell C 65-70)			
Rail Roughness	Less than or equal to 1.6 µm			
Maximum Protrusion off the Rails (X and Y face of satellite)	8.5mm +/- 0.1mm			
Maximum Mass	12kg			
Center of Mass	+/- 8cm from geometric center			
RBF/ABF	Remove or Apply Before Flight feature must be physically accessible via deployer access ports on one of the X faces			
Deployment Switches	Must have at least 3 separate captive deployment switches that correspond to independent electrical inhibits			
Deployable Systems and Integration Constraints	Deployable systems must not rely on deployer for deployment			
Deployment Velocity and Tip-Off Rate Compatibility	CubeSat shall be capable of withstanding 0.5 to 1.5 m/s of deployment velocity			
Operation	Must not be operating in any way while in the deployer			
EPS Connection	No more than 6 inches of wire between power source and electrical inhibit			
Interfaces	There should not be any interfaces between the satellite and the deployer			
Environment Interface Requirements				
Acceleration Loads	+/- 7.0g (Nx)	+/-4.0g (Ny)	+/-4.0g (Nz)	+/-13.5 rad/sec^2 (Rx,y,z)
Random Vibration	Survive tests for 60 seconds 20-2000Hz			
On-orbit Acceleration	0.2G, During Airlock Carry Out 1.5m/sec^2, During E-STOP Maneuver 500mm/sec^2, 12 deg/sec^2			
Integrated Loads Environment	Withstand a force of 1200N across all load points equally in the Z direction			
Airlock Depressurization	104.8 kPa to 0 kPa at 1.0kPa/sec			

important requirement as the debris orbiting earth has already become a problem. If this is ignored, then we could effectively be trapped within a mine field and limit future space exploration.

2.3 Trade Studies

The trade study process outlines the way that all design decisions, especially those relating to hardware, are made. The purpose of this is meant to systematically tell us what the optimal decision is for the mission, first in terms of objective constraints, and next in terms of subjective constraints. The flow of a generic hardware trade study can be seen in figure 3. Usually this process takes about two weeks and is fully outlined in a file per trade study, with the primary time constraint being the time it takes to get quotes for each candidate. First, we define objective constraints based on the needs of the mission and then supplement those with subjective constraints, which are usually more pertinent to talent and resources existing to optimally handle the hardware. Next, in the survey of industry stage, the contributing lab members contact companies and find all possible options that have a chance to be utilized. Usually the target is to focus on components with a technology readiness level of 9. The TRL metric can be seen in table 9 and has been used as a metric for space technology readiness since it was established in the 1970s by NASA [42]. The path to making that final decision lies with quantitative ranking, first on objective constraint satisfaction, then on subjective constrain satisfaction. If there are more than

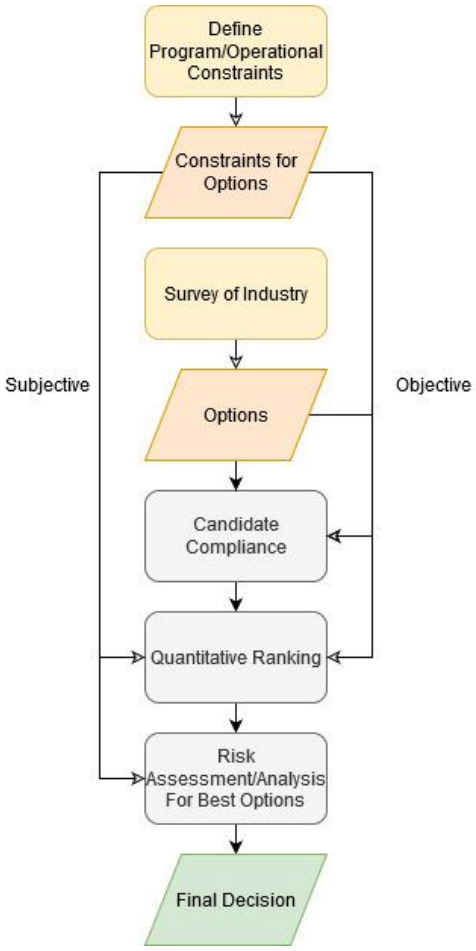

Figure 3 - Trade Study Flow Diagram

Table 9 - NASA Technology Readiness Level Definitions

TRL-1	Basic Principles Observed and Reported
TRL-2	Technology Concept and/or Application Formulated
TRL-3	Analytical and Experimental Critical Function and/or characteristic proof-of-concept
TRL-4	Component/subsystem validation in laboratory environment
TRL-5	System/subsystem/component validation in relevant environment
TRL-6	System/subsystem model or prototyping demonstration in a relevant end-to-end environment
TRL-7	System prototyping demonstration in an operational environment
TRL-8	Actual system completed and “mission qualified” through test and demonstration in an operation environment
TRL-9	Actual system “mission proven” through successful mission operations

one “optimal” decision available, in depth risk analysis is conducted to decide on the final option.

In the following sections, the major categories of subsystems that comprise the stack are discussed. The command and power subsystems are discussed first as they provide the basis of operation for the satellite. That will be followed by communications as that facilitates external control and collection of data. Finally, the payload and data collection related electronics will be discussed, acting as the method for facilitating the purpose for the mission.

2.4 Electronic Hardware

Hardware and electronics teams work together in the process of developing MOCI’s electronics stack. The trade studies completed for many of the components were done more than two years ago. Now, hardware team primarily focuses on the design and testing of interface boards while the electronics team focuses on the implementation of flight and embedded software to support those components. Students working in leadership and advisory roles over the course of the past few years have played vital

Table 10 - Major Electronics Personnel Acknowledgments

Name	Individual Contributions	Position
Jackson Parker	Payload Software, Embedded Payload Software	Systems Engineer
Caleb Adams	Payload Software	Program Manager
Alex Lin	Flight Software, Payload Software, Testing and Integration	Will become Systems Engineer
Godfrey Hendrix	Lab IT, Payload Software	Electronics Lead
Justin Heimerl	Payload Interface Board Design	Hardware Lead
Allen Spain	PCB Design	Hardware Advisor
Eric Miller	Embedded Payload Software, Flight Software	Electronics Team Member
Alex Holmes	Embedded Payload Software	Will become Electronics Lead
James Roach	Flight Software	Previous Electronics Lead
Austin Kinkade	Payload Interface Board Design	Hardware Team Member

roles in the progression of electronics portion of the satellite design to its current state. The most notable of these individuals are included in table 10 with their most notable contributions.

The term “stack” should be taken literally when discussing a CubeSat like MOCI. All boards are literally stacked on top of each other and interfaced vertically through a PC104+ interface. These connectors’ form factor make them perfect for stackable CubeSat systems as they fit on interface boards that sit within a 10x10cm (1U) space.

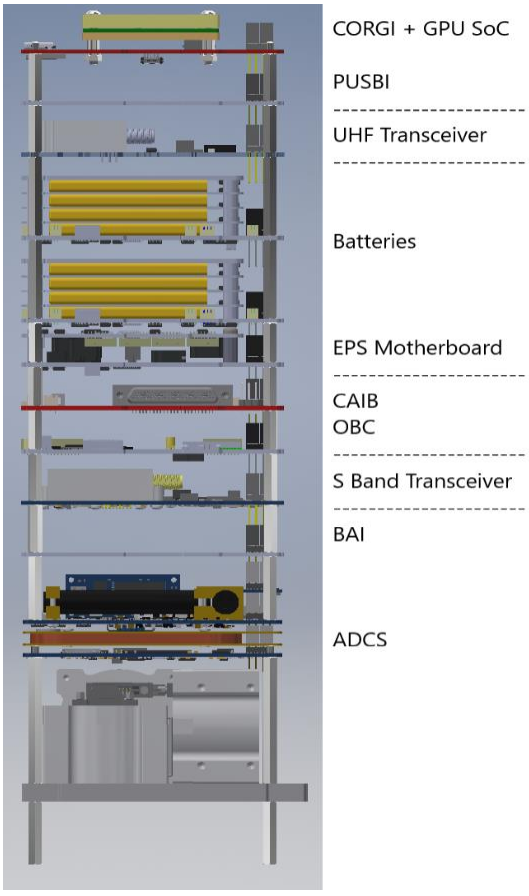

Figure 4 - MOCI's Isolated Stack Labeled

Many COTS subsystems in the small satellite industry, especially from companies targeting CubeSat developers, have adopted this type of connector. With similarities in bus pinout as well as configurability through option sheets, users are not constrained to ordering an entire bus from one manufacturer and can pick based on the mission's subsystem needs. Even if the pinouts are different, the development of a very simple interface board for rerouting is all that is necessary. The PC104 interfacing can be seen on the right side of figure 4 which shows the labeled stack.

2.4.1 Core Avionics Hardware

Core avionics hardware is made up of the on-board computer (OBC), electrical power system (EPS) and communications subsystems. Together, these systems provide the foundation for nominal satellite operation, ignoring the necessity to perform experiments. The Clyde Space OBC can be considered MOCI's brain. It manages the state of the satellite through the monitoring of telemetry, scheduling and command of all subsystems. Software running on the OBC is considered flight software and is discussed further in section 2.5. This system utilizes a Smart Fusion 2 FPGA SoC that has a Cortex-M3 processor delivering 62.5 DMIPS. As this is the most critical component of any mission utilizing it, levels of failsafing exist on the OBC including hardware level error detection and correction (EDAC), triple modular redundancy registers and safe state machines. These are primarily implemented to protect against radiation related upsets. Watchdog and latch-up protection circuitry also exist to reconcile anomalous behavior outlined by lack of response or a high current reading. In order to allow the OBC access to non-Clyde Space components, like ISISpace UHF deployment control, the core avionics interface board (CAIB) was developed. The primary purposes of this board include electrical

ground support interfacing, inhibit actuation, acting access point to EPS, and direct connection to UHF antenna for deployment.

The EPS handles the power management and distribution of the satellite. This includes the EPS motherboard, the batteries, and the solar panels. MOCI is utilizing the Clyde Space's XUA EPS motherboard and their 80Whr battery system, but the hardware team will be designing custom solar panels that utilize the CESI CTJ-LC2 solar cells that have a form factor of 26.5 cm^2. The EPS is directly interfaced with the battery, which comes in the form of two 40Whr batteries stacked, for monitoring and power distribution. They can effectively be considered one system after they are connected. During operation, the EPS utilized has a variety of failsafe conditions that lead to the entire bus shutting down until the condition has returned to normal. Examples of these include battery-under-voltage, under-current, over-current and over-voltage. Watchdogs also exist in order to reset and reconcile any issues within the EPS microcontroller. When considering safeguards that the batteries have themselves, mechanical inhibits are included on the batteries when delivered and the batteries have heaters in preparation for operation in a space environment. The necessity of the heaters is due to the storage and operating temperature of the batteries being at -10 degrees Celsius, higher than any other component on MOCI. Other than cost efficiency, designing our own solar panels gives us a high degree of freedom for configuration. Even though solar panel design primarily entails routing the leads from the solar cells, the panels must be designed around entirely mechanical concerns, which is different than most custom PCBs developed for MOCI. This is the last real design process left in MOCI and requires the finalization of the ADCS mount as most mechanical concerns and keep out zones are

due to the ADCS, and where its components are mounted. Another careful consideration, as outlined in the CubeSpace ADCS User Manual [10], is the necessity to remove any possibility for current loops on solar panels. If one does exist due to poorly planned routing, it can cause a magnetic moment in the direction of the sun potentially causing the satellite to tumble.

MOCI's power generation and storage requirements that facilitated the decisions on the EPS component trade studies were primarily from the power budget. This was created to determine if a given configuration would be "power positive", meaning there exists enough power generation and storage capacity to operate. To build valuable margin into the power budget, we must consider worst case circumstances for power generation, power draw from subsystems and storage capacity of the batteries. Due to MOCI's targeted orbit, the sun will not always be available. Utilizing the STK orbit simulator, makes the process of estimating power generation over a certain period much easier. This simulation is configured by providing the solar cell efficiency, worst case number of cells per panel and orbital parameters. The data generated from this simulation is fed into an excel spreadsheet that details the worst-case power draw for all the components as well as the capability of the batteries to hold the charge that the cells are generating. Parameterization of MOCI's current power budget assumes the following operations over a one-year period:

- Data downlink for 4 minutes once a day
- One SfM pass per week (including slew maneuver and imaging)
- 20 minutes of data processing after each pass
- elsewise always in Idle - Cruise mode

Even though most of MOCI's solar panels will have 5-6 cells, the power budget simulation was run with a nominal configuration of only 4 solar cells per panel to ensure that MOCI will have a safe margin in terms of power generation. To account for cell degradation and unknown factors that would affect solar cell efficiency, this simulation was run 4 times with power generated reduced by 5% each time [49]. As seen in table 11, MOCI will remain power positive under the proposed schedule to a maximum of 10% hit to panel efficiency. This result inspires a large degree of confidence that MOCI will be able to operate unrestricted by power limitations..

Table 11 - MOCI's Power Budget Results Pre-PIR

% Generation	Mean Batt (Whr)	Min Batt (Whr)	Max Batt (Whr)
100	71.12657066933765	26.605994999992284	80
95	67.60224759926571	16.429634583325832	80
90	62.61244841246279	4.077789999991901	80
85	62.41324786069563	-0.027182916674147427	80

The communication subsystems represent the last portion of the core avionics hardware. There are a few options for communications in with each option has a specific frequency range. The trend being that higher frequencies have the higher data rates. This is also true for power consumption, which can really narrow the possibilities for certain types of communication on systems like CubeSats. The International Telecommunication Union has bands ranging from 3kHz (VLF or very low frequency) to 3THz (THF or tremendously high frequency). As described in NASA's state of the art small satellite technologies, the frequency range utilized for ground to spacecraft communication generally ranges between 30MHz and 40GHz, which make up the VHF through SHF ranges. This list of communication bands can be seen below in table 12 [56].

Table 12 - Spacecraft Communication Frequencies

VHF	30 - 300 MHz
UHF	300 MHz - 3 GHz
L Band	1 - 2 GHz
S Band	2 - 4 GHz
C Band	4 - 8 GHz
X Band	8 - 12 GHz
Ku Band	12 - 18 GHz
K Band	18 - 27 GHz
Ka Band	27 - 40 GHz
Optical (Laser Communication)	100 - 800 THz

MOCI is utilizing UHF for uplink, telemetry beaconing and repeating. This is an ideal frequency for these tasks as they do not require the transmission of large amounts of information, so slower data rates are allowable. The added benefit with this type of communication is that it usually is implemented with low directionality or omni directionality, which allows for communication with the satellite without the satellite pointing in a specific direction. The specific system that MOCI is utilizing is an F'Sati UTRX transceiver paired with a turnstile deployable UHF antenna system from Innovative solutions in space. This combined system is capable of half duplex omni directional communication from 430-440MHz. The communication task that does require a significant amount of data, which would be unreasonable for the UHF system to attempt to transmit, is experiment data downlink. S Band is being utilized for this purpose, utilizing the F'Sati STX transponder transceiver and S Band patch antenna for transmitting

between 2.4 and 2.45 GHz. This system differs from the UHF system as it is directional and only meant for downlink, with the directionality being defined through the patch's 60-degree field of view. A link budget was evaluated for both S Band and UHF communications considering all antenna and ground station properties. This link budget and the communication related portions of the data budget are used to verify that these systems and the ground station allow for data downlink and schedule up linking that need to occur during mission operations. It has been confirmed that the current configuration in terms of ground station and communications systems on MOCI provides margins as outlined in table 13.

Table 13 - MOCI Link Budget Margins

UHF Uplink Margin	36.4 dB
UHF Downlink Margin	24.6 dB
S Band Downlink Margin	5.8 dB

2.4.2 Optics

The Ruda Cardinal optical train represents MOCI's primary payload. Much of this work must be credited to Mathew Hevert, a previous mechanical team member and now an employee at Ruda Cardinal. Matthew and his coworkers have developed an unbelievable optical train, capable of both RGB and grayscale imaging, with low cost being heavily attributed to his work on the project. This system has gone through a few significant changes throughout the development of MOCI and led to MOCI moving from a 3U to a 6U back in Fall of 2018. The major changes to the system and growth in size was primarily to obtain the highest spatial resolution that could be afforded with the already purchased system. This metric is characterized by the imagers ground sample distance (GSD), which is the spatial size of each pixel at a certain orbit. The GSD

calculation for the RGB and Grayscale sensors can be seen in table 14 utilizing the equations below, along with the information about the individual sensors in table 15.

$$GSD = \frac{altitude * pixel\ size}{focal\ length}$$

$$FOV = 2 * \arctan\left(\frac{sensor\ width}{2 * focal\ length}\right) * \left(\frac{180}{\pi}\right)$$

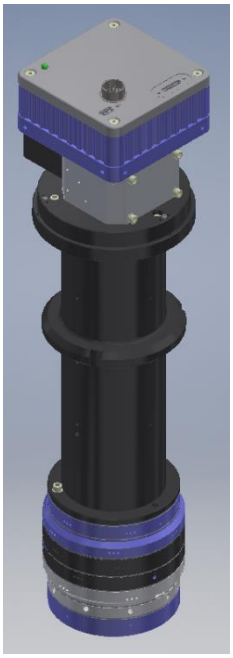

Figure 5 - MOCI's Isolated Ruda Cardinal Optical Train

Table 14 - GSD Calculations

Altitude (km)	GSD – Imperx (m)	GSD – Blackfly (m)
400	6.67	8.68
425	7.08	9.22
450	7.50	9.77
475	7.92	10.31
500	8.33	10.85
525	8.75	11.39

Table 15 - MOCI Camera Sensor Specifications

Sensor	Imperx C4180		Blackfly	
Number of Channels	1 (grayscale)		3 (RGB)	
Pixel Size	4.5E-6m		5.86E-6m	
Bit Depth	8	10	10	12
Max Frame Rate (USB3 – 5Gb/s)	49.67	39.74	217.014	200.38
Image Size	4096x3072		1920x1200	
Field of View	3.91x2.93		2.38x1.49	
Focal Length	2.70E-1m			
Aperture	75.6mm			

2.4.3 Attitude Determination and Control System

A propulsion system on MOCI is completely unnecessary as it will never require any immediate change in orbit during its mission lifetime. Due to MOCI's requirement to latitude-longitude point track and nadir point, the utilization of an attitude determination and control system is necessary. For CubeSats, the systems are generally made up of similar types of actuators and sensors. Sensors include magnetometers, rate sensors, star trackers and other cameras/light sensors for tracking earths horizon or the sun.

Actuation is accomplished using reaction wheels and/or magnetorquers. The combination of sensory input and complex momentum exchange methods allow the ADCS unit to slew the satellite, utilizing complex control system methods that are out of the scope of this thesis.

For the satellite to be able to image a portion of the ground numerous times during one pass, a requirement for MO-1, the ADCS system must be able to keep the field of view (FOV) of the optics system overlapping during that time. The number of images, viewpoint differences and overlap between images for MO-1 to be fulfilled optimally was studied early on in mission concept development by Caleb Adams and Nicholas Neel [8]. The process of collected SfM images entails a symmetric slew over the target as the satellite passes overhead, with the entry and exit angles both being the same off nadir. A simple example of this for visualization purposes is provided in figure 6. Generally, a higher percentage of image overlap, more images and larger differences in viewing angle lead to better results. Determining what slew rate is necessary to take overlapping images at a specific frame rate within this maneuver requires a simple geometry problem, referred to as the pointing budget. A pointing budget for a usable maneuver can be seen in table 16.

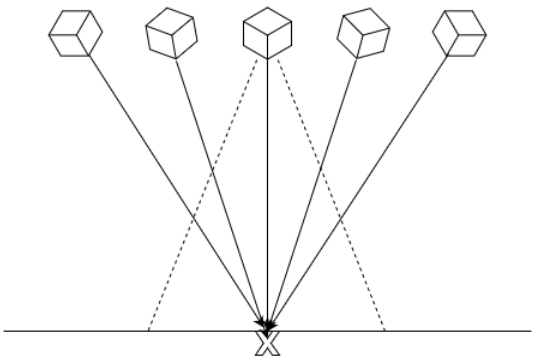

Figure 6 - Visual Example of SfM Point Tracking Slew

Table 16 - Example MOCI Pointing Budget for SfM Maneuver at 400km

Maneuver Parameters	Attitude	400 km
	Entry and Exit Angle	15 degrees
	Minimum % Overlap	50%
	Number of Images	30
	FOV constraint	3 degrees

Maneuver Requirements	Slew Rate Required	2.285955467 degrees/sec
	Frame Rate Required	0.4374538413 fps
	Allowable Boresight Error	0.05237184314 km

After the initial trade study was conducted, MOCI chose to utilize the Adcole Maryland MAI401 as its ADCS system. Better options for this system existed that would have provided less error and higher viewpoint angle differences, but the MAI401 fit within MOCI's budget and fulfilled the requirements for SfM slewing. The optimal choice would have been one of the Blue Canyon XACT ADCS systems, but those cost more than $120,000 which was outside budgetary allowances. Unfortunately, as MOCI was in the process of its hardware acquisition phase, MAI decided to cancel the product line. This led to the trade study being reopened in the fall of 2019, where no choices from the original survey of industry fell within the budget. After surveying the CubeSat ADCS industry again, the CubeSpace ADCS was discovered; a snapshot of which can be seen in figure 7. Even as our only choice, this system has flight heritage as well as worst case attitude determination and pointing accuracies less than a tenth of a degree. The primary downgrades that resulted from this change included a lower star tracking rate and a necessity to mount the system ourselves. The CubeSpace star tracker has a max tracking rate of 0.3 degrees per second which means that accuracy for pointing will decrease the longer MOCI is slewing faster

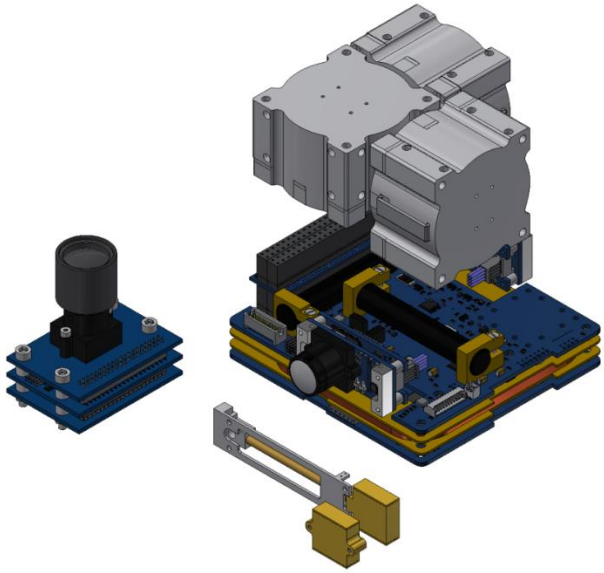

Figure 7 - CubeSpace ADCS Components

than that, as the rate sensors will be the primary method of attitude determination. Regardless of the change in decision, sensor refresh rates are usually 1 Hz for many of these systems. This either minimizes the maximum imaging frame rate to 1 fps or requires camera parameter estimation be added to the MOCI pipeline before launch. After discussing our maneuver with the CubeSpace team, due to the short time span of the SfM maneuver the rate sensors should not lead to error that would break the pointing budget. Also, worth mentioning, due to the CubeSpace PC104 pinout not lining up with MOCI's master pinout, an interface board had to be developed. The BAI (Board ADCS Interface) is an extremely simple two-layer board that simply reroutes the PC104 pinout of the CubeSpace ADCS to conform with MOCI's.

2.4.4 Payload Interface Boards

The two boards designed by electrical engineering students at the SSRL for payload interfacing include the CORe GPU Interface Board (CORGI) and the Peripheral USB Interface board (PUSBI). As could be assumed, the CORGI acts as the carrier board for MOCI's payload processor, NVIDIA's Jetson TX2i, which will be discussed more in the next chapter. This board represents the most complex design that the hardware team has been working on. The purpose of the board is to interface the TX2i with the rest of MOCI's stack as well as provide debug and testing interfaces to ensure the TX2i is properly configured and functioning once mounted. The interfaces included in the version of CORGI that will be utilized on MOCI include a Micro SD card port, USB 2.0 Micro port, 2 USB 3.0 Type A ports, PC104+ header, and the TX2 connector. Due to the TX2 modules not having software enabled power circuitry built in, MOCI's hardware team had to develop this circuitry for addition on the CORGI. This is the primary contributor to the complexity of the design and requires very accurately timed signals, requiring extensive

simulation during the development process. For debugging, the USB ports will be utilized. For flashing, the USB 2.0 Micro port will be utilized. To obtain internet connectivity during debugging, an ethernet to USB converter will be attached to one of the USB 3.0 ports, but will be utilized for connecting MOCI's cameras when this is unnecessary.

The PUSBI is a simpler board that acts only to interface the optical system with the CORGI over USB 3.0. Even though this board's design is simple, if not done properly it represents a single point of failure. For mission assurance purposes and the radiation sensitivity of GPUs, this interface board also includes USB to SPI converters to enable the possibility of the OBC to command the optical train. This is done to ensure that MOCI will be able to acquire and downlink images even when the payload processor stops functioning. Due to the OBC not being able to handle reading images at the same speed as USB 3.0 transmits (5 Gb/s), the converters only utilize the USB 2.0 lines of the cameras.

2.5 Flight and Embedded Software

After discussing the electronics that make up the entire satellite, it only makes sense to discuss the software that makes the operation of MOCI possible. This comes in two forms: embedded payload software and flight software. The embedded payload software executes on the payload processor in a Linux environment to process commands sent from the OBC as well as command and internally monitor the payload processor. Flight software (FSW) is executed on the OBC to command and monitor all subsystems, including the payload. Building a spacecraft flight software framework from the ground up is a lengthy process and a tall order to fill, so both of SSRL's missions utilize the GenerationOne flight software development kit (FSDK) by Bright Ascension

(BA). This is a proprietary framework developed specifically for the small satellite and nanosatellite market. The BA FSDK facilitates the development, modification, and testing of flight software; this is carried out with a more object-oriented approach where the software is encapsulated into “components” that are defined using a combination of C source and XML. These components represent both hardware and software subsystems carrying out mission-critical functions that can be added and removed easily to a generated software deployment for execution. This system is so lightweight that once generated for a set of specific hardware, the system can run on something as simple as a Raspberry Pi or other embedded systems. The FSDK can be utilized on operating systems like Linux, FreeRTOS, and RTEMS or as a standalone system operating solely on its built-in cooperative multi-tasking capabilities. With the Clyde Space OBC having a native port of FreeRTOS compiled for execution on its processor architecture, this is the operating system that MOCI is utilizing. A real time operating system has multiple benefits over a traditional general-purpose operating system, such as timely response to events and interrupts without requiring significant CPU resources. The use of this does however require a higher level of technical knowledge and familiarity with RTOS systems to develop on. Two different BA FSW deployments are used on the OBC: one for the onboard failsafe image and one for the primary image. A primary image contains all the components and drivers necessary for full operation, while the failsafe image is stripped to only allow barebone operation and critical components (OBC, EPS, UHF communications) and is generally only booted into for significant debugging or error.

2.5.1 Command and Data Handling

Due to MOCI being in LEO, the satellite will not be constantly visible for ground operators to communicate with the satellite. Even if we could monitor the satellite's state constantly, the latency in commanding the satellite from the ground would be too great for reliable command of the satellite. Due to this a command and data handling system within MOCI's FSW is necessary to sustain safe and nominal operation without direct guidance. As the OBC acts as the brain of the satellite where the FSW is being executed, it acts as the symbolic, and in most cases literal, master to the subsystems that make up the stack. Except for the EPS and payload, all components act as I2C slaves to the OBC. A high-level diagram showing the data interfaces can be seen in figure 8.

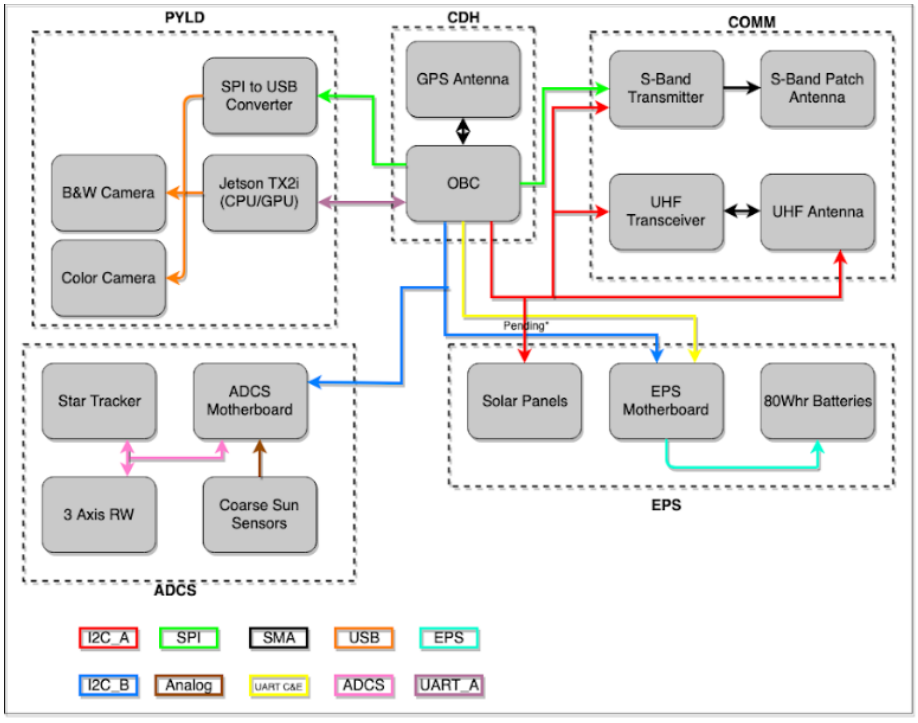

Figure 8 - MOCI's Communication Interface Block Diagram

MOCI's FSW data handling system facilitates the monitoring, processing, logging and downlinking of all telemetry and data on the entire satellite. The built-in sampler, logger, and monitor templates provided by BA facilitate allow for FSW developers to implement components to handle telemetry for specific subsystems. This allows both the mission operators to monitor the state of the satellite through constant telemetry beacons over UHF, as well as allow the OBC to monitor for specific subsystem telemetry changes that necessitate automated action. Commands sent from the OBC to other subsystems are highly automated, and utilize BA's action templates with triggers like time, event and orbit. MOCI will only be utilizing time and event actions as orbit actions are an unnecessary complication for this mission. The series of commands and data handlers used at a given point in operation depend on MOCI's high level mode, discussed in section 2.5.3.

2.5.2 Core Avionics and Payload Components

One of the largest benefits to utilizing this FSDK is the long list of platform support packages that contain proven drivers for small satellite TRL-9 level COTS subsystems. The subsystem components defined in the FSW deployment facilitate the OBC's ability to command and monitor the entire satellite. The platform support packages provided by BA encompass most of the stack and are easily included to MOCI's deployment with additional initialization and configuration. The components that are not included in platform support packages are the ADCS and the payload drivers, which must be developed in-house. The ADCS driver, despite a large and thorough software interface, is quite simple as CubeSpace already has this interface defined as various I2C commands. The latest FSDK includes a unit tested Cube Wheels component as part of the suite, but additional implementation is required to utilize the rest of the ADCS's

functionality. Payload components, as they are completely custom, necessitate a full design of the API

The payload processor API utilizes UART for communication and has currently proven functionality in its first version through flatsat testing. A list of programmatic states, which can be seen in figure 9, were implemented as FSW commands on the OBC, while they were implemented as programmatic state scripts in Python on the payload processor. When the OBC commands the payload processor to enter a certain state, the payload will execute the corresponding programmatic state script that then invokes a system call to perform the actions defined by that state. During any of these high-level states the OBC can request data/telemetry or interrupt processing. To ensure that the OBC has command of the payload processor and no anomalous behavior is allowed acknowledgement transactions are incorporated into all commands.

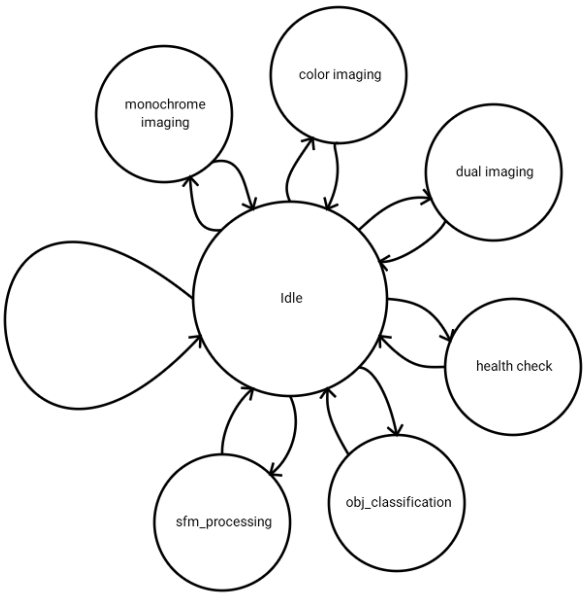

Figure 9 - MOCI Payload Processor Programmatic States

Due to the possibility of the experimental payload processor failing after a few months to a few years of operation, failsafe/redundant payload related FSW must be developed. The primary components necessary to mitigate this include the processor death verification protocol and the redundant camera drivers. The processor death protocol is a series of steps the OBC will take every time the TX2i fails to respond. Once the OBC fails to communicate with the payload processor after multiple attempts the OBC will invoke a hard power cycle by commanding the EPS to cut power and then attempt to communicate again. After a few attempts at doing this, the OBC will pronounce the payload processor as “dead” and permanently cut power to it, a simple diagram can be seen in figure 10. The redundant camera drivers give the OBC the ability to command the cameras and take images in the case of processor death. This is a comparatively difficult task as the OBC does not have USB interfaces (only UART, I2C, and SPI), so a custom driver must be implemented for USB to SPI conversion. This driver is the most difficult that MOCI has to develop and due to its redundancy, it is not a complete priority right now, so development on this will not be complete until PSR.

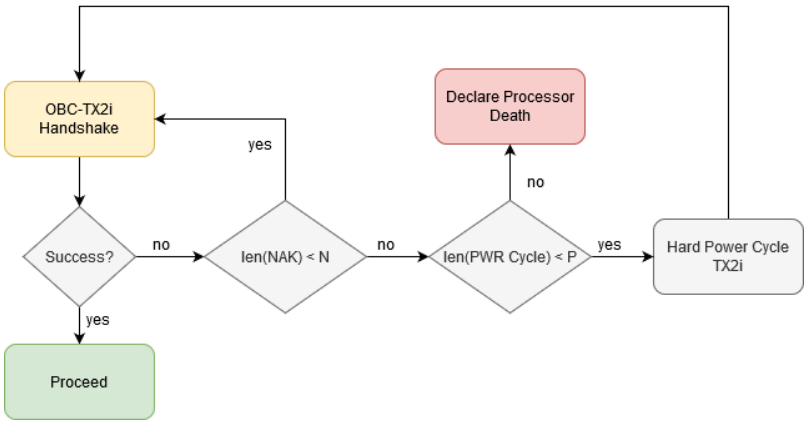

Figure 10 - MOCI Payload Processor Death Confirmation

2.5.3 System-level Software Components

System-level software components, such as the mode manager and scheduler, guide the high-level operations of the satellite. As defined in the concept of operations deliverable, and seen in figure 11, the possible modes and transitions between them allow for the actual execution of MOCI's experiments as well as robust and safe operation. The mode manager is a custom component that handles the execution of modes as well as the transition between them. While in a given mode, a corresponding finite state machine is utilized to guide operation within that mode. For the purpose of transitioning in and out of modes, each mode has entry and exit criteria. When a mode change is requested, the tasks in order to fulfill the exit criteria are completed. This ensures that modes cannot overlap and cause anomalous, or possibly dangerous, operation. To guide mode transitions, ground operators make use of the mode manager's scheduler by up linking packetized schedules, with each entry having a TimeAction item and mode parameters. The TimeAction is a component provided by BA that allows for both absolute- and relative-time based execution and function calling. When a TimeAction entry's given time matches the onboard time, the mode manager retrieves the mode to transition to as well as the associated parameters. The FSW deployment also leverages event-driven automation components that monitor important parameters, such as battery voltage and overall temperature: when these parameters fall outside a designated safe range, these components cause the mode manager to recover to safe mode immediately.

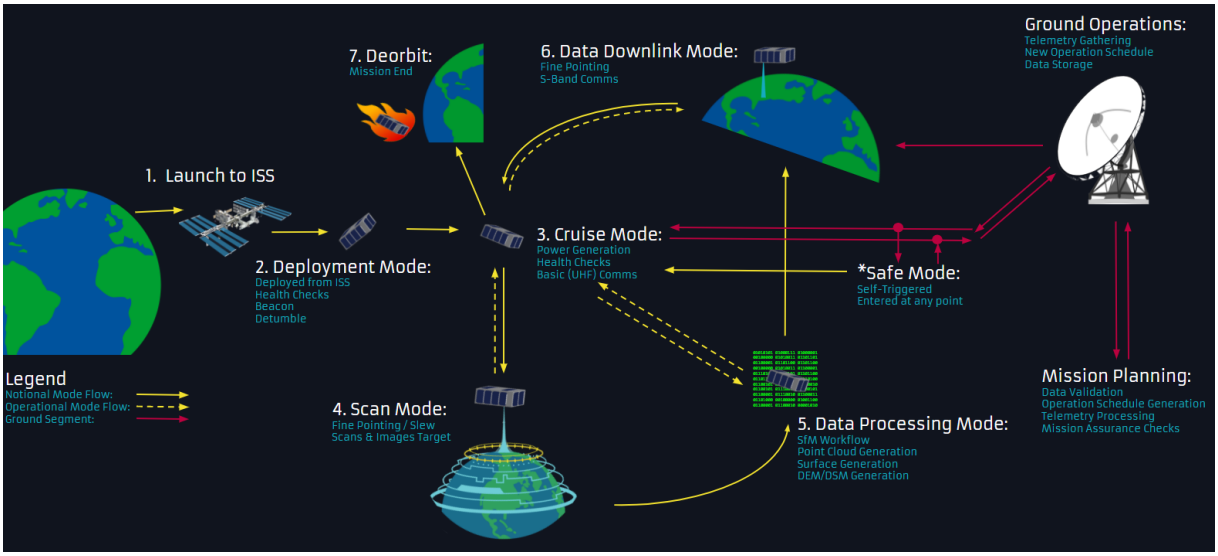

Figure 11 - MOCI Concept of Operations

2.5.4 Embedded Payload Software

Other than the API discussed in section 2.5.2, it is also important to mention some embedded software that was necessary to implement on the payload to mitigate operational concerns. As a safe measure, implementation of signal handling and checkpointing within the payload experiment software is necessary. This is to allow for incremental payload processing incase transition out of data processing mode is necessary during computation. Next, as the GPU SoC generates a significant amount of heat during processing, the mechanical team leads requested a GPU burn program to act as a heating mechanism. This program, which may later be added as a programmatic state for the payload processor API, is a simple repetitive execution of a vector addition GPU program to keep temperature from dropping out of a desired range while power levels allow execution. Finally, even though the majority of MOCI's data products will be in the form of compressed ply files, image downlink will be requested for object recognition data products and verification. This requires the use of image compression to minimize the size of the images as much as possible while still keeping most of the information.

Two options exist and will be tested on MOCI, one being the lossless LZW algorithm [20] and the other being the lossy JPEG2000 algorithm [21]; both of which are known to be state-of-the-art for their respective type of compression. Embedded software native to the payload processor utilized on MOCI is discussed in section 3.2.2.

2.6 Mechanical

Regardless of how robust the electronic and software design of a satellite is, if structural and other mechanical considerations are dismissed in the slightest bit the possibility of mission failure goes up exponentially. During the design process for a small satellite like MOCI, the mechanical team's high-level considerations are:

- Designing and fabricating the frame
- Considering full system structural integrity
- Discovering and mitigating thermal and radiation concerns
- Planning integration

The mechanical team members that had the most meaningful contributions to this portion of the mission can be seen in table 17.

Table 17 - Major Mechanical Personnel Acknowledgments

Name	Individual Contributions	Position
Michael Ely	Structural Design and Analysis	Mechanical Team Lead
Clay Busbey	Structural Design, CAD, Manufacturing	Will be Mechanical Team Lead
Grayson Bellamy	Structural and Thermal Analysis, CAD	Mechanical Team Member
Nicholas Heavner	Structural Design, Pointing Budget	Previous Mechanical Team Lead
Matt Hevert	Optomechanical Design and Analysis	Previous Mechanical Team Member
Graham Grable	Design and Mission Conception	Previous Chief Engineer
Casper Versteeg	Thermal and Structural Design + Analysis	Previous Mechanical Team Lead

2.6.1 Structure

As mentioned, MOCI's mechanical team has been designing its own frame to be utilized with MOCI. This provides a few significant benefits mostly surrounding cost reduction and higher levels of flexibility given during stack organization and assembly. Due to the large list of mechanical variables present the use of the ANSYS simulation package finite element structural analysis is necessary when designing the frame and other structural components. The primary goal of this simulation and careful consideration is to ensure that MOCI can withstand the large amount of stress that occurs during a launch. This does not just mean preventing total structural failure, but also preventing the ADCS or optical train from being misaligned and/or damaged. Due to these being very sensitive components to mechanical shift, the success of MOCI relies heavily on mechanical stability. In order to provide optimal freedom of motion and flex to relieve potentially harmful stresses in the optical train, RUDA cardinal has worked very closely with our mechanical team to design an optimal mount. The most difficult part about mounting the ADCS system is the reaction wheels. Currently mechanical team members are iterating on designs with oversight from CubeSpace to ensure that alignment will be ensured. The main goal and way to mitigate the possibility of misalignment is to find a good way to mount all wheels to the same piece of metal. The method of utilizing ANSYS to determine the structural integrity has been to add fixed supports like MOCI will see within its launch deployer and then performing modal characterization and inertial loading to determine the natural resonant frequency and deformation acceleration from launch. The previously mentioned requirement of the 100 Hz fundamental frequency minimum

was confirmed with the frame alone being at 625.42 Hz and the frame with the ADCS and optical train being at 1311.1 Hz. It was also confirmed that at 35 Gs of loading, deformation was negligible.

In addition to designing the frame, MOCI's mechanical team will also be performing most of the fabrication for the frame and brackets using UGA facilities. Careful consideration during the design process was necessary in order to make this possible as resources and tools are not like that of a large-scale manufacturing facility. The material being used is 6061-T6 aluminum and most of the work done with this material be done in UGA's Driftmier Engineering Center machine shop on a CNC 3-axis HAAS MiniMill. Other parts will be manufactured by the UGA Instrument Shop and other 3rd-parties such as Team Metal Finishers and Plethora. The manufacturing of each part begins by determining the size requirements so that a piece of raw stock aluminum can be cut to size. This raw stock is rough-sawn larger than the dimensions of the final part and varies based on the types of milling and work holding being used. Most parts will have rough stock cut to be 0.25" larger than the part sides and 0.1" larger on the top and bottom. This stock will be held in a CNC vise and milled in two operations, top and bottom. Some parts will require more operations and more complex work holding, such as soft jaws, gauge pin, and toe clamps. The most difficult to machine parts will be the rails as they have many different square corners that can only be created by multiple operations in multiple orientations.

An example of the machining simulation of one of the side panels can be seen in figure 12. This is necessary because it allows us to plan the order of operations and create the tool paths that the CNC will read. Each of the blue lines is a mill cutting the aluminum

to create the parts geometry. For validation, each manufactured part is thoroughly analyzed through measurement with calipers, micrometers and gauge pins for deviations from the computer-generated model, then corrected.

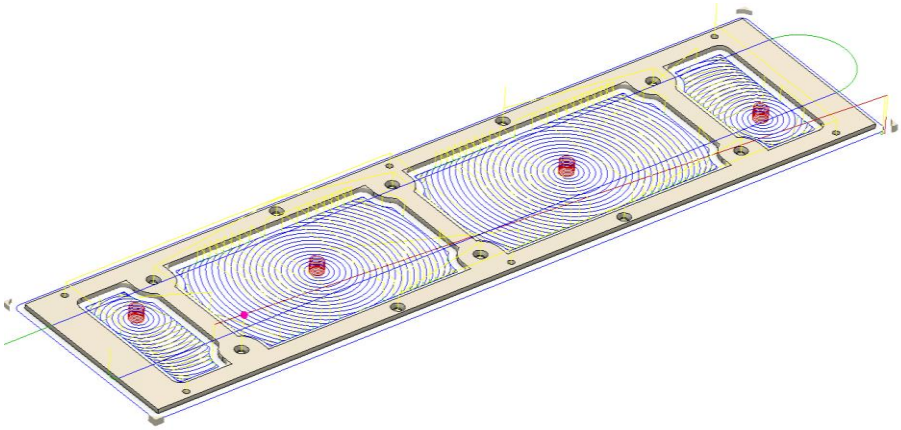

Figure 12 - MOCI Side Panel Machining Simulation

2.6.2 Thermal & Radiation Mitigation

Due to the lack of convection in space, accounting for conductive heat transfer and minimal radiative dissipation is necessary. Using Thermal Desktop and MATLAB simulations, written by mechanical team members for verification, the thermal profile of MOCI during operation is analyzed. The primary purpose of this is to ensure that electrical components being used on MOCI do not leave their storage or operating temperature range in the respective situation. As the simulations currently stand, no component leaves its storage temperature when it is off, and no component leaves its operating temperature when on. Usually satellites have a harder time dealing with hot cases as there is not much in the way of dissipating heat in space. In order to ensure that the hot case is mitigated, the mechanical team has employed a series of heat straps and heat sinks to give hot components like the payload processor a way to dump heat into the frame.

The mechanical team has also found a few ways to shield parts of the satellite system from radiation. The two major ways that radiation is being shielded from components on MOCI is from thick aluminum and use of Dunmore SatKit shielding [31]. The primary shield being utilized on MOCI is the aluminum block meant to surround the GPU SoC on the CORGI interface board. This both acts as a heat sink and a layer of protection from ionized particles. The Dunmore SatKit is a thermal shielding kit that includes two layers of STARcrest™ MLI materials and polyimide tape. Neither of these fully mitigate the radiation concerns, especially with high energy particles, but they do reduce the frequency of the effects.

2.7 Integration

Even though the electronics and hardware team play a significant role in the assembly process of the satellite, the mechanical team plans the process far in advance. Preparation for integration includes the development of the mechanical ground support system (MGSE) as well as fully defining the assembly procedures that will be conducted in the SSRL's 100,000-particle rated cleanroom. The MGSE is simply used for locking the satellite into place and ensuring that nothing but the rails have pressure applied to them. Two versions of this exists, the horizontal and vertical MGSE. Both utilized in different portions of the assembly procedures. The assembly procedures carefully outline every addition to the stack. This includes all the torque specifications for anything that needs to be screwed into place and any necessary specific ordering. It must also be noted that this majorly mechanical process must make room for incremental testing to ensure that electronic interfacing occurs properly.

The major responsibility of the electronics and hardware team during and after the integration process includes the testing and monitoring of satellite functionality. To do this, we utilize a custom electrical ground support equipment (EGSE) system made of a Raspberry Pi, power supply, laptop and HackRF One connected to the satellite with a custom interface board [48]. This system enables the following to occur within the partially or fully integrated satellite: battery charge/discharge, power distribution, inhibit actuation, command and data handling and RF communications.

2.8 Testing

The SSRL utilizes a Clyde Space flatsat, a custom thermal vacuum chamber (TVAC) and is currently developing a structure from motion test rig for PIR. The flatsat provides an invaluable way for FSW developers to test the satellite in a flat configuration. This flatsat has eight PC104 headers that are routed together to act as if the elements were stacked. This system is the primary way that testing for PIR is being facilitated, as it allows us to prove communication between the subsystems as well as a simulated operation that would occur in orbit. The TVAC chamber provides an analogous benefit by allowing the SSRL to test the functionality and durability of systems in a space-like environment. After pumping down the pressure to 1E-6 Torr, we can thermal cycle components by using the built-in heating elements that can bring ambient temperature up to 80C and liquid nitrogen that can bring ambient temperature to -50C. Lastly, the SFM test rig is being developed to prove the functionality of the payload research software while being connected to the flat sat. By placing an object on the rig, a series of images will be taken while the object rotates, providing the payload processor with data analogous to what it will be acquiring in orbit.

2.9 Mission Operations

Before launch, the Mission Operations Team is in preparation mode. As stated by UNP in their user's guide, the mission planning portion is the most neglected stage of design as most of the time leading up to launch has been spent on satellite design, integration and testing. As with the other three teams, there were some individuals that have been invaluable contributors to the work necessary to plan MOCI's future. Their names and contributions can be seen below in table 18.

Table 18 - Major Mission Operations Personnel Acknowledgements

Name	Individual Contributions	Position
Mateen Saki	Ground Station Architecture and Testing	MOPS Team Lead
Sydney Whilden	Simulations and Concept of Operations	Previous MOPS Team Lead
Conor Mcferren	Simulations and Scheduler	Will be MOPS Team Lead

2.9.1 Licensing

Due to MOCI having communication subsystems and plans to transmit and receive RF power, it requires an FCC experimental license. Due to the utilization of amateur bands, we must also acquire International Amateur Radio Union (IARU) coordination. Also, due to MOCI including two imagers it must acquire a Commercial Remote Sensing Regulatory Affairs (CRSRA) license from the National Ocean and Atmospheric Administration (NOAA). To obtain an FCC license, both the IARU coordination and NOAA license must be acquired first. The current progress on licensing includes IARU coordination being confirmed and the plan to contact NOAA before May of 2020. The primary reason why NOAA has not already been contacted is due to the requirement of an orbital debris assessment report (ODAR). This report is generated by NASA's debris assessment software that takes a master materials list of the satellite. As design changes have concluded, the process of acquiring this should be on the horizon.

2.9.2 Post-launch operations

Once the satellite has been confirmed to have a successful launch and deployment, satellite operation begins, and the structure of the lab changes slightly. Generally, at this time, systems engineers are replaced with one or more flight directors that organize the operation of the satellite and attendance of ground operators. Ground operators are tasked with monitoring telemetry beacons, uploading schedules, and being present for data downlinks. Currently a large part of the preparation for the mission's future is with the development and testing of the UGA SSRL's ground station, which can be seen in figure 13. In order to prepare for future missions or collaborations, it was built with more than just the capability of communicating with SPOC and MOCI. The ground station has full tracking capabilities with a pointing accuracy of about 0.1 degrees and made up of UHF and VHF Yagis as well as a large S Band dish, with specifications listed in table 19.

Figure 13 - Image of SSRL Ground Station

Table 19 - UGA SSRL Ground Station Specifications

System	Beam Width (degrees)	Gain (dBic)	Frequency Range
UHF Yagi	21	18.9	430-440 MHz
VHF Yagi	38	14.39	144-148 MHz
S Band Dish	3.6	33	2.4103 GHz

CHAPTER 3: DESIGN APPROACH FOR PROCESSOR PAYLOAD

3.1 Description of Hardware Platform

3.1.1 GPU vs. CPU platforms

The differences between a traditional Central Processing Unit (CPU) platform and a Graphical Processing Unit (GPU) platform explains the necessity of a specialized payload processor and steps taken in order to utilize it for MOCI's experiments. At the highest level, the largest difference between the two is the number of cores both have. This is of course an oversimplification. If it wasn't then the use for CPUs would have been completely replaced by GPUs. Take a step down towards what a modern CPU and GPU core look like, their use cases and true difference are seen.

CPUs were the first computer processors capable of arbitrary command execution based on an instruction set, the idea being derived from the universal Turing machine. Through four generations of computers capable of this, the concept of a CPUs primary components has remained the same, arithmetic logical units (ALUs), processor registers, and a control unit. Improvement of each of these components as well as hierarchical memory implementations have made CPUs extremely efficient at performing sequential tasks. With the emergence of microprocessors as the fourth generation, multiprocessing became a commonality due to CPUs containing multiple cores. Today, CPUs are extremely capable and highly dynamic computational units, their main benefit over GPU cores. However due to the CPU core being more complex, they are much more expensive fiscally and spatially. This is why CPUs cannot beat GPUs in their parallelization abilities.

At first, GPUs were simple specialized processors that helped a CPU articulate a display from information. Starting with simple vector machines that were utilized in early computers and arcade games, the GPU evolved to be able to handle more and more information. These processors have always been meant for large scale SIMD processing, in other words processors capable of executing the same instruction on many different pieces of data simultaneously. In the beginning they were utilized specifically for graphics, which encouraged a lot of advancement due to incredible consumer need for video games. Once video game consoles were introduced, game developers stopped being the hardware designers, making it desirable to have more capability for general purpose utilization of the hardware. This is when the OpenGL API was developed, allowing hardware designers to start designing around developer requirements. The increasing consumer need for GPU technology, primarily in gaming, eventually led to the development of general-purpose GPU processing (GPGPU) frameworks, like CUDA and OpenCL. These frameworks led to an explosion in utilization for a variety of applications like artificial intelligence, general software acceleration, scientific model computation and much more. Today, what companies like NVIDIA call CUDA cores are just ALUs. Due to their simplicity, they are extremely cheap fiscally and spatially, allowing a large amount of them to be organized into clusters for SIMD processing. Therefore, GPUs are used for parallel processing while CPUs are used for general purpose tasks.

3.1.2 Impact of use on a Small Satellite Platform

The reason that MOCI represents a technology demonstration is due to the use of a GPU SoC as an onboard processor. The primary benefit of this technology can be measured in GFLOP per dollar as well as GFLOP per man hour in programming and

developing an interface. The added computational ability that a GPU provides on a small satellite, without much of a noticeable increase in cost, make it an extremely easy and powerful COTS addition. With the added benefit, there is still a reason that this is a technology demonstration. GPUs are inherently not designed to function optimally in a space environment. Primary reasons why these are not commonly used pieces of hardware on spacecraft include concerns including heavy power draw and heat generation during processing as well as the general susceptibility to radiation that comes with components that are so close together. The robustness of the hardware chosen and designed to help utilize this technology was described in the last chapter. The ways robustness was built into the operation and software of MOCI's hardware accelerator is described in part through the rest of this chapter.

3.2 Justification of NVIDIA as hardware of choice

3.2.1 Hardware Accelerators

Hardware accelerators that have been used for computer vision include graphics processing units (GPUs), field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs). Noticeable trends in state-of-the-art hardware tend to show the tradeoffs in utilizing one of these types of accelerators, as depicted in figure 14. For MOCI's purpose of demonstrating a small and cheap system capable of high-performance computation a GPU was the most innovative and relevant hardware choice. This decision was made easier due to the required level of expertise to utilize and FPGA or ASIC on a system, as well as the cost associated with their utilization [15]. Also, the much higher level of programmability that GPUs provide allow for testing and acceleration of multiple different computations depending on the data received. This is a trait that is

especially necessary for MOCI to accomplish the execution of the pipeline for generating DSMs.

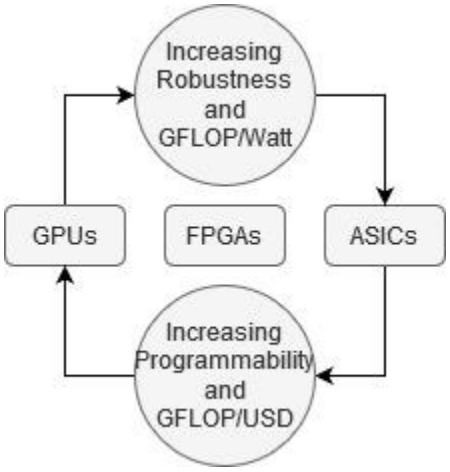

Figure 14 - Hardware Accelerator Trends

The utilization of GPUs for image processing specifically has been occurring since the release of the general-purpose GPU processing frameworks CUDA and OpenCL around 2007. Most notably, this technology has led to the rapid expansion and improvement of machine learning methods, like deep learning. A common method of doing this, is the utilization of the neural network or more specifically convolutional neural networks (CNN) for things like object recognition and segmentation, both being done on MOCI. The convolutions involved in these networks are incredibly repetitive and very easily parallelized, so many CNNs greatly benefit from the SIMD processing capabilities of GPUs. The emergence of general-purpose GPU SoCs has given even more power to this concept, leading to MOCI's choice in hardware.

SoCs are generally defined as an integrated chip that contain all components necessary to be considered a computer. everything necessary to use the system as a standalone computational system. These can exist in many forms and often include a wide range of peripherals, have a small form factor and consume less power than their

non-SoC counterparts. For MOCI's purposes, GPU SoCs are very attractive. Due to the complexity of interfacing a GPU chip on a board, not to mention other implications like higher power draw, a unified solution is highly desirable. Primary developers of SoCs with GPUs on them include Qualcomm, AMD and NVIDIA. Qualcomm has offered embedded units, like its SnapDragon line, for high performance smart phones, but have recently released a robotics platform called the RB3, unfortunately too late for MOCI's consideration. AMD has publicized usage of its embedded platform in aerospace application and has chips utilized on few Unibap boards that also contain FPGAs. These are generally great solutions, but most require interfacing memory to the SoC and do not actually come as a complete standalone system. The MOCI mission is not looking to develop a board meant for production and market, we are looking for a solution that can be utilized and interfaced by the talent we have at hand. The NVIDIA Jetson series provides that solution as a complete standalone system with a simple interface, becoming the choice for MOCI's payload processor. This allowed use of the better-established GPU development framework, CUDA, and tools that come with the unit further described in the next section.

3.2.2 NVIDIA Jetson TX2i

NVIDIA's Jetson series in general has been utilized extensively for robotics, automation and computer vision tasks since the initial release of the TK1 in 2014. The small footprint and lower power consumption of these high-performance processors provide a distinct advantage for embedded projects. Modules of this family have been utilized in a variety of notable projects like the Nintendo switch gaming system and the Skydio 2 drone. Since the TK1, the Jetson series has grown to include the TX2, Nano and Xavier platforms with more likely to come. Major differences in these modules can be

found in the GPU architectures, the amount of memory, available peripherals and associated level of power consumption. However, they are all SoC's, meaning that CPU's and GPU's are combined on a single PCB and share LPDDR4 memory. This provides distinct advantages in terms of the system acting as a full embedded solution and the offering of a zero-copy capable GPU-CPU processing system. In most GPU computations, the CPU must first transfer memory into the GPU's dedicated RAM and then request the updated memory once the processing has completed. This is one of the most notable bottlenecks in GPU processing. With zero-copy, the unified RAM allows the capability of removing this bottleneck, which is very advantageous in video or image-streaming applications. The primary differences in the SoC's among the NVIDIA Jetson series can be seen in table 20, as well as prices listed throughout NVIDIA's website as of February 2020.

MOCI is utilizing the TX2i module, the TX2 module version designed for use in industrial environments. The primary workhorse of all the TX2 modules is the Parker System on a Chip (SoC). This SoC includes an NVIDIA Pascal GPU that has 2 Streaming Multiprocessors each having 128 CUDA cores in it, two NVIDIA Denver 2 CPU cores and 4 ARM Cotrex-A57 CPU cores. In section 2.4.4 the TX2i interfaces being utilized were discussed. Some peripheral interfaces offered by the TX2i that are not utilized by CORGI include: HDMI, DisplayPort, PCIe, SPI, CAN, I2C, and more. Future iterations of the board will likely be more comprehensive, but as an initial technology demonstration, there was no need to include more than what was necessary for debugging and operating on the stack.

Table 20 - NVIDIA Jetson Modules

NVIDIA Jetson Module	CPU	GPU	Price (on arrow.com if not available on nvidia website)
TK1	Quad-Core ARM Cortex-A15	192-core NVIDIA Kepler	no longer available
TX1	Quad-Core ARM Cortex-A57	256-core NVIDIA Pascal	$368.00
TX2	Dual-Core NVIDIA Denver 1.5 64-Bit + Quad-Core ARM Cortex-A57		$479.00/module
4GB TX2			$299.00/module
TX2i		512-core NVIDIA Volta w/ 64 Tensor Cores	$789.00/module
XAVIER NX	6-core NVIDIA Carmel ARMv8.2 64-bit	384-core NVIDIA Volta w/ 48 Tensor Cores	$399/module
8GB AGX XAVIER			$679.00/module
AGX XAVIER	8-core NVIDIA Carmel Armv8.2 64-bit	512-core NVIDIA Volta GPU with 64 Tensor Cores	$699.00 for module and development board
Nano	Quad-Core ARM Cortex-A57	128-core NVIDIA Maxwell	$99 for module and development board

The reasoning behind the decision to use the TX2i version over the standard TX2 led to the tradeoff between robustness in more extreme environments versus a slight decrease in performance. The primary differences and similarities between the TX2 modules can be seen in table 21, compiled from the official NVIDIA Jetson TX2 Module datasheet [18]. As can be seen, the TX2 module has a slight advantage when it comes to overall power usage and the peak memory bandwidth of the LPDDR4 memory controller. However, due to the modules requirement to survive a launch and operate in a space environment, the increased temperature range, vibration rating and inclusion of ECC are worth the slight loss in performance. ECC specifically will be very helpful during operation in a space environment due to the likelihood of data corruption from SEEs caused by radiation. This comes in the form of inline ECC stored in 512 bytes of LPDDR4 and has single bit error correction and double bit error detection [18].

Table 21 - NVIDIA Jetson TX2i Module Differences

Module	TX2	4GB TX2	TX2i
Memory (128-bit LPDDR4)	8GB 59.7 GB/s	4GB 51.2 GB/s	8GB with ECC Support 51.2 GB/s
Storage (eMMC Flash)	32GB	16GB	32GB
Size (section 6.5)	50x87x7.15mm	46x83x7.2mm	
Module Power	7.5W (Max-Q) / 15W (Max-P)		10W (Max-Q) / 20W (Max-P)
Power Input	5.5V – 19.6V	9.0V – 19.6V	
Shock	140G, 2ms		
Vibration	10Hz – 200Hz, 1g & 2g RMS		Random and Sinusoidal: 5g RMS 10Hz – 500Hz
Temperature Range	-25C to 80C		-40C to 85C
Expected Operating Life	5 years		10 years
Other Environment Testing	N/A		Free Fall, Mixed Gas Flow, Dust Effects

Aside from the hardware, the JetPack SDK for each of the Jetson modules provide an incredible jump start to help developers maximize the utilization of their hardware. The package's primary libraries and APIs include CUDA, TensorRT, cuDNN, Multimedia API, VisionWorks and OpenCV. Developer tools include Nsight Eclipse Edition, CUDA-GDB, CUDA-MEMCHECK, Nsight Systems, nvprof, Visual Profiler and Nsight Graphics. Valuable command line tools like NVPmodel and tegrastats also exist for management and monitoring of the SoC's internal resources. NVPmodel is extremely useful as you can

turn off CPU cores and set maximum operating frequencies for both the CPU and GPU. For the TX2i, any of the modes shown in table 22 can be utilized, and if users would like more options, they can configure their own model. To keep performance of the TX2i high as well as lower overall power consumption to around 10W, MAX-Q mode is being utilized on MOCI.

Table 22 - NVPmodel Modes

Mode Name	Denver Core Utilization	ARM Core Utilization	GPU Frequency
MAX-N	2 Cores @ 2.0 GHz	4 Cores @ 2.0 GHz	1.30 GHz
MAX-Q	N/A	4 Cores @ 1.2 GHz	0.85 GHz
MAX-P Core-All	2 Cores @ 1.4 GHz	4 Cores @ 1.4 GHz	1.12 GHz
MAX-P ARM	N/A	4 Cores @ 2.0 GHz	1.12 GHz
MAX-P Denver	1 Core @ 2.0 GHz	1 Cores @ 2.0 GHz	1.12 GHz

The easiest way to get all these features on the Jetson modules is through the installation of Linux for Tegra (L4T), a variant of Ubuntu with all the JetPack components included. As could be assumed, MOCI will not need all the packages previously listed once in orbit, much like many other production level applications of this hardware. With the use of the NVIDIA SDK manager, only packages that are strictly necessary for the operation of MOCI and execution of its experiments are being utilized. Instead of using the L4T as the operating system, MOCI is using Yacto [17] to build a minimal operating system with a custom U-boot bootloader for triple modular redundancy at the OS image level. The system is currently in development but is expected to include a RAM file system, meaning when the TX2i is on it will never write to flash memory to reduce any degradation.

3.2.3 CUDA

The ability to program and utilize the GPU for parallelization outlines its entire purpose within MOCI, other than to command the images when operable. This is done through the implementation of CUDA [11], a C/C++ framework, at its lowest level. This language is what the popular AI frameworks like TensorFlow have as a backend allowing the use of GPUs for optimization. CUDA and OpenCL are the primary GPU programming frameworks right now and have been since their release. Both are very similar with subtle differences in their hardware utilization and thus threading schemes. Even though CUDA only works on NVIDIA hardware, unlike OpenCL, it has always been the leading framework. This is primarily due to the extensive CUDA toolkit containing quite a few useful libraries to help beginners getting started and prevent experienced CUDA programmers from reinventing the wheel.

A CUDA function and GPU functions in general are denoted by the term kernel and are written like any other function. The main difference lies in the fact that this kernel will be executed across the device simultaneously with differentiation in memory access allowable by using block and thread identifiers. The general flow starts with the CPU preparing and transferring data to an accessible location for the GPU. Once the data is prepared for GPU access, the CPU portion of the program calls the kernel with the proper block and grid dimensions relative to the amount of information being processed in parallel. Each individual process, or thread, on the GPU can determine what data to reference by looking at its xyz block and grid identifiers. During the time the GPU is processing, the CPU continues and will only wait if a global thread fence is called or a transfer of the memory back to the CPU is called. The way this is mapped onto hardware is simple and can be seen in figure 15. The CUDA scheduler assigns a single block to an

SM with each SM having access to the global GPU memory allocated as well as its own volatile shared memory cache for its own threads to utilize. An interesting feature of CUDA and NVIDIA hardware is noted when considering the encapsulated SPs within the SMs. Scheduling of instruction execution occurs in warps mapped onto SMs consisting of 32 threads executing in lockstep, also seen in figure 15. Having knowledge of this is useful for optimization, as execution speed can be increased by ensuring that branches in the same warp do not have branch divergence to allow for full SIMD execution. Another important consideration when developing a CUDA kernels is that each thread has its own memory where variables that cannot be shared by other threads are stored. If more

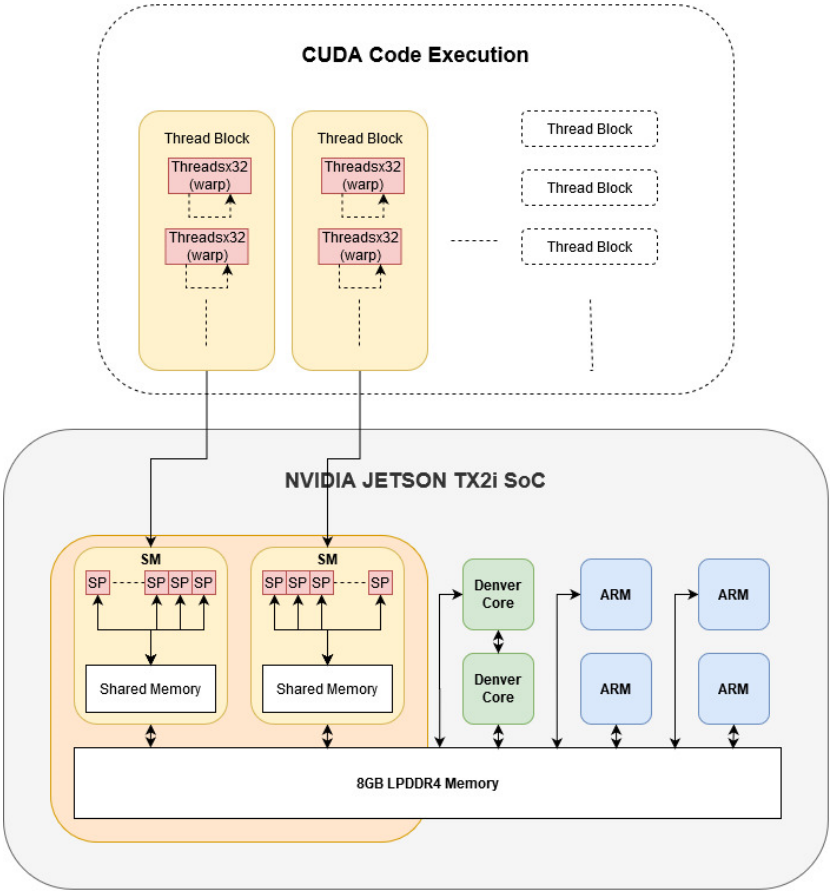

Figure 15 - TX2i Architecture Overview and Code Execution

memory is needed than allowed, the kernel does not fail, it just overflows into global memory significantly reducing memory access speed.

3.3 Work required to design and implement payload experiment software used in this project

There have been five primary contributors to the development and testing of MOCI's payload software framework. Their names and primary contributions are listed in table 23.

Table 23 - Major Payload Experiment Software Contributions

Name	Contribution
Jackson Parker	Design and Implementation of Foundational Framework, Data Structure Design, Feature Detection, Matching, Object Detection, Data Parallel Octree, Data Parallel Quadtree, Marching Cubes,
Caleb Adams	Conceptualization of the Pipeline, Point Cloud Generation and Bundle Adjustment
Godfrey Hendrix	Cloud Segmentation and TensorRT Conversion
Alex Lin	Poisson Mesh Solution and Final Data Product Generation
Jake Conley	Unit Testing and Blender Simulation

3.3.1 Image Acquisition

Image acquisition occurs when the satellite is in scan mode. As with all modes other than deployment and safe mode, the satellite enters this mode based on a schedule defined by the ground operators and uplinked while MOCI is in orbit. The submodes that make up scan mode can be seen in table 24. To ensure certain targets are being imaged, the mission operations team carefully plans transition into scan mode knowing MOCI's orbital parameters and when its propagation intersects with the target list. For SfM data acquisition, targets have been picked based on a guarantee that there will be enough elevation change to be visible in images taken by the 6.67m GSD RUDA cardinal optical train. Targets currently on the list for this purpose are primarily mountains, including but

not limited to Mount St. Helens, Mount Shasta, Telescope Peak, Mount Rainier. The targets purposed for object recognition are being expanded based on the proven capabilities of the CNN and the existence of labeled datasets. Currently a few inclusions in that part of the list are the Sapelo Island, Sydney Harbor, University of Georgia, Golden Gate Bridge, and the Charleston Harbor.

Table 24 - MOCI Scan Mode Submode Details

SCAN MODE	
Submodes	**Description**
Point Prep	Satellite orients to configured pointing mode, boots payload (CMOS camera and TX2i), loads previously uploaded acquisition parameters from the TX2i.
Acquire Images	Target Pointing: slew begins, and MOCI executes image acquisitions, and then ceases to slew.
	Nadir Pointing: MOCI maintains normal orientation and MOCI executes image acquisitions. All images are tagged as they are acquired.
Point Exit	Satellite turns off payload and transitions back to Cruise mode.

The major differences between the scans meant to acquire data for either SfM or object recognition is the mode that the ADCS system is in. For SfM related imaging, the ADCS will be in latitude-longitude target tracking mode. This necessitates action during the point prep and more importantly point exit submodes as this maneuver is short but requires rapid slewing and could lead to tumbling that would need to be rectified. Object recognition on the other hand does not require much action during these submodes due to the ADCS staying in nadir tracking mode, the same mode as when the satellite is in cruise mode. When taking images for the purpose of either SfM or object recognition, the association of images with ADCS telemetry is desired for both camera parameter estimation and GPS localization of detected objects. Due to the lack of a direct line of

communication between the ADCS and payload processor and potential latency when allowing the OBC to control image captures, imaging needs to be executed based on a predefined time configuration. The current implementation of this configuration of this can be seen in figure 16 and defines a certain number of images to be taken with a discrete time interval. Possible implementation of a dynamic configuration, where time between images can vary during the series capture, may be included on MOCI but has been currently deemed unnecessary. During this schedule, the payload processor will be commanding the acquisition of images while the OBC coordinates the delivery of ADCS telemetry as close to the schedule as possible. Due to the CubeSpace ADCS's sensor refresh rate of 1Hz, ADCS telemetry can only be associated with an image once every second. If captures are faster than that camera parameters estimation or interpolation between the ADCS telemetry would be necessary. Due to difficulty with synchronization based on the OBC's RTOS and other subsystem priorities, it will not be assumed that there is perfect synchronization between ADCS telemetry acquisition and image captures. However, these values do not need to be perfect for either of the data processing submodes and will be used as initial guesses for camera parameters before filtering and adjustment occurs.

Time of First Image	Number of Images	Time Interval

Figure 16 - Image Capture Timing Configuration

3.3.2 Goals

The overall goal of the payload experiment software developed for the MOCI mission is to generate data products when the satellite is in data processing mode. The submodes that outline specific action within this mode are listed in table 25, where the SfM and Object Recognition submodes make up the meat of data product generation.

During SfM, DSMs will be generated. Labeled images will be generated during object recognition. At a system level the difference between these submodes is minimal as the OBC passes the same parameters, a target directory for processing and program parameter string, to the payload processor during the request to enter the respective programmatic state. Providing this functionality with as minimal and robust of a system as possible is important. This means having fine control over both memory allocation as well as signal handling, while also having very accurate and configurable algorithms for acquisition and simple onboard adjustment towards optimal results on board. This means we cannot utilize OpenCV, even though it represents the most comprehensive computer vision framework available. Implementations of SfM, as well as object detection and classification are all possible using OpenCV with many GPU accelerated methods available. However, the level of abstraction and the lack of some notably accurate and more complicated GPU accelerated algorithms like SIFT, make the framework bloated and incomplete for our purposes. Due to this, I have developed a custom computer vision framework with a focus on GPU acceleration and modularity to suit the needs mentioned above as well as facilitate easy collaborative efforts; which is very important due to the turnover associated with a research lab that is primarily made of talent

Table 25 - MOCI Data Processing Submode Details

DATA PROCESSING MODE	
Submodes	**Description**
Compute Prep	Turn on the TX2i and establish communication with the TX2i control program.
Structure-from-Motion	Using acquired imagery, data is processed through SfM pipeline
Object Recognition	Detect, localize, and identify non-land target objects

Post-Processing	Post-processing - compresses, error-encodes, packetizes, and stores the final data products
Compute Exit	Commands the TX2i to gracefully save and exit any running programs and turn off.

3.3.3 SSRLCV

SSRLCV is a custom C++/CUDA computer vision framework that designed for current and future collaborations. The current version of SSRLCV provides simple methods that facilitate the abstraction of GPU memory transactions, simple methods for GPU accelerated image manipulation, and usage of GPU accelerated state-of-the-art computer vision algorithms. For future proofing collaborations, the organization of the large code base is split into factories that utilize and produce the primary data structures in SSRLCV. The current components of these factories as well as important utility structures will be discussed throughout this section.

The first and most important custom data structure within SSRLCV is Unity. The purpose of Unity is to abstract away overly verbose CUDA memory transactions as well as provide a set of utility functions that help manipulate an array meant for CUDA processing. This class is utilized in every factory as a wrapper for input and output arrays, so its general functionality should be understood. Implementation of this class is in the format of a header only library due to its templated nature. Its general design and inclusion of methods is to have some resemblance to the vector structure offered by the C++ standard library, as this structure is well known and should help C++ users become comfortable with its use quicker. The major benefit of Unity is that it houses pointers to both host CPU and

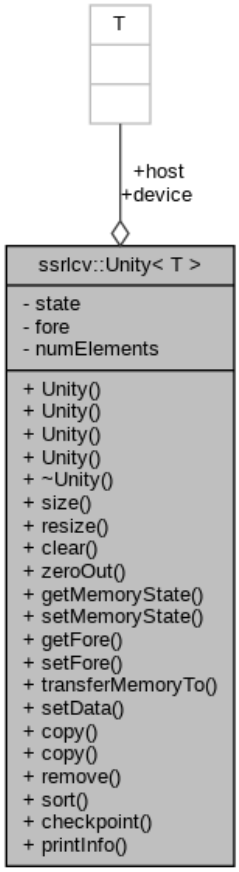

Figure 17 - SSRLCV Unity UML Diagram

device GPU memory. This allows the user to utilize one variable that holds both GPU operable and CPU operable memory. The state variable, an instance of the MemoryState enumeration seen in table 26, allows the user to determine what state a given Unity instance is in and to allow methods with a Unity argument to eliminate state assumptions. Memory transaction methods are the basis of functionality of Unity, mainly including hard and soft transfers between the two memory locations as well as clearing memory in one or both locations. Of course, the user can take care of these processes themselves, but by just setting the fore variable to the location of the most updated memory Unity will take care of all transactions and ensure that a specified state transition happens properly. Other methods within Unity are for simple data manipulation like sorting and stream compaction. Also, due to its usage throughout SSRLCV Unity was also the clearest location to write the checkpointing methods mentioned in section 2.5.4. To accomplish this, binary read and writes were implemented that force type checks to ensure that the file is compatible when reading. The signal handlers implemented in the main.cpp files of pipelines call this to ensure progress is kept and later if the file exists, a Unity can be recreated from the binary uty file.

Table 26 - MemoryState Enumeration

State	Unity<T> State Assumption
null	size = 0, device & host = nullptr
cpu	device = nullptr but data exists in host
gpu	host = nullptr but data exists in device
both	data exists in both host and device
pinned	memory exists in both host and device with host being allocated as CUDA pinned memory
unified	memory exists only in device bust can be accessed by either CPU or GPU with CUDA unified memory

The Image class represents the primary wrapper for pixel and camera information within SSRLCV, which most factories take instances of. Currently this class contains a Unity<unsigned char> as its pixel information, for ease in prototyping early portions of the framework. Soon the Image class and its variables will be templated to include any arithmetic type as a pixel. This is allowable due to the class primarily acting as a wrapper and not having type specific methods. Outside of the class, currently implemented GPU accelerated methods to manipulate pixel information include bilinear interpolation scaling, border addition, normalization, convolution, pixel type conversion, color depth conversion and gradient calculation.

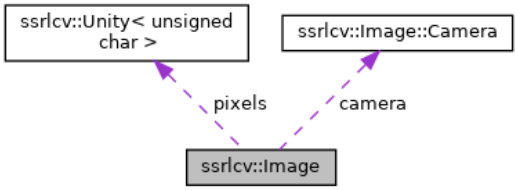

Figure 18 - Image Class Graph

The first major factory to discuss is the FeatureFactory. As there are a wide variety of feature detection algorithms that produce different feature descriptors from provided pixel information, it made sense for this class to act as a parent class, including methods that could be utilized to improve any feature detector. The child classes produce descriptors in the form of templated Feature structs, seen in figure 19, where the template is the descriptor. The first child FeatureFactory implemented within SSRLCV was the SIFT_FeatureFactory. This class implements the well-known Scale and Rotation Invariant Feature Transform developed by David Lowe [25]. The SIFT feature detector and its variants are known as the best and most accurate detection algorithms

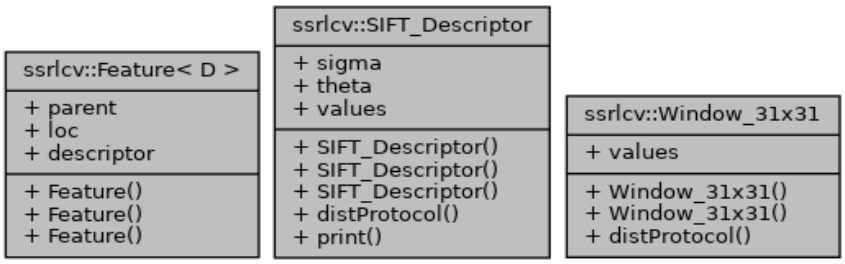

Figure 19 - UML Diagrams for Feature (left), SIFT_Descriptor (center), Disparity Window Descriptor Example (right)

due to the verbosity of the 128-dimensional descriptor, seen in figure 19. The implementation of this algorithm also provided the first two contributions to the FeatureFactory parent utility methods. This came in the form of the scale space implementation that was formulated in the algorithm for scale invariance, and the primary orientation determination algorithm for rotation invariance. With these portions of the SIFT feature detection algorithm implemented in the parent FeatureFactory class, now any new child of FeatureFactory can make its features scale and rotation invariant. A UML class graph of this relationship as well as a class graph for the ScaleSpace container structure

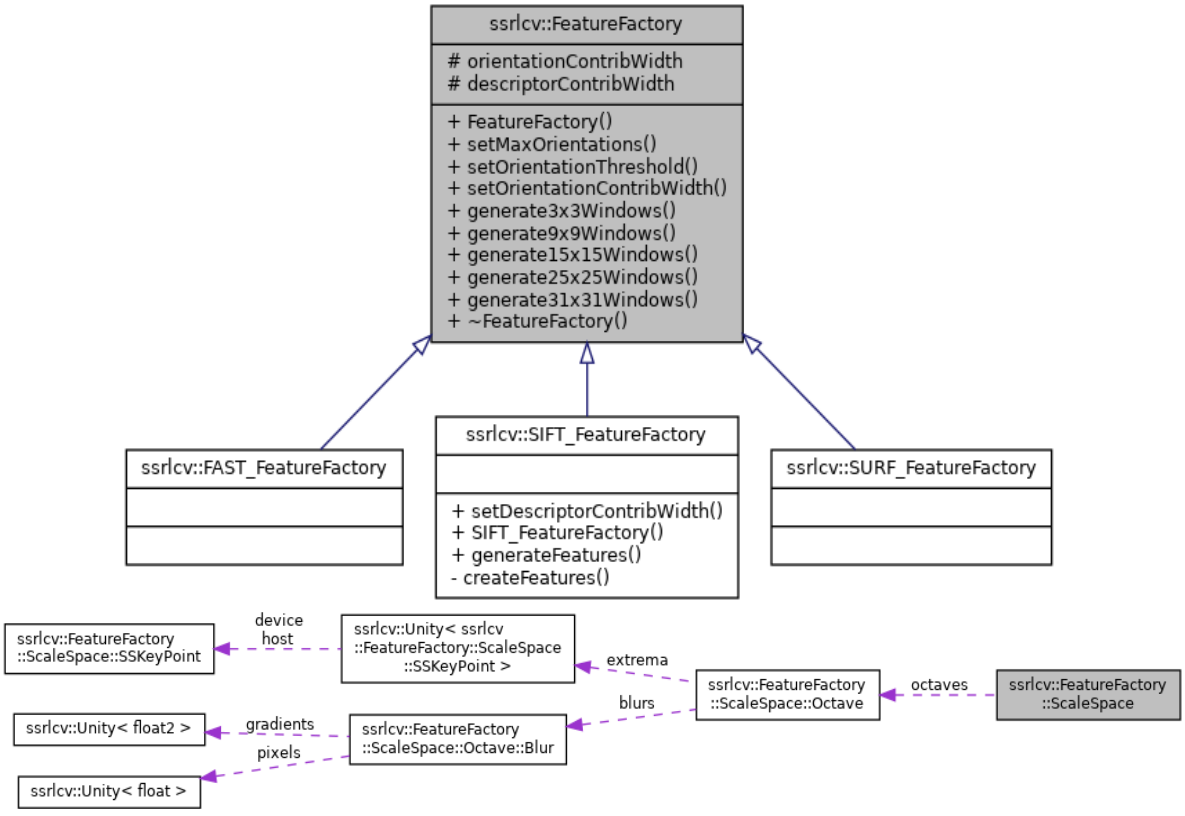

Figure 20 - FeatureFactory with Inheritance UML Diagram (top) FeatureFactory::ScaleSpace Class Graph (bottom)

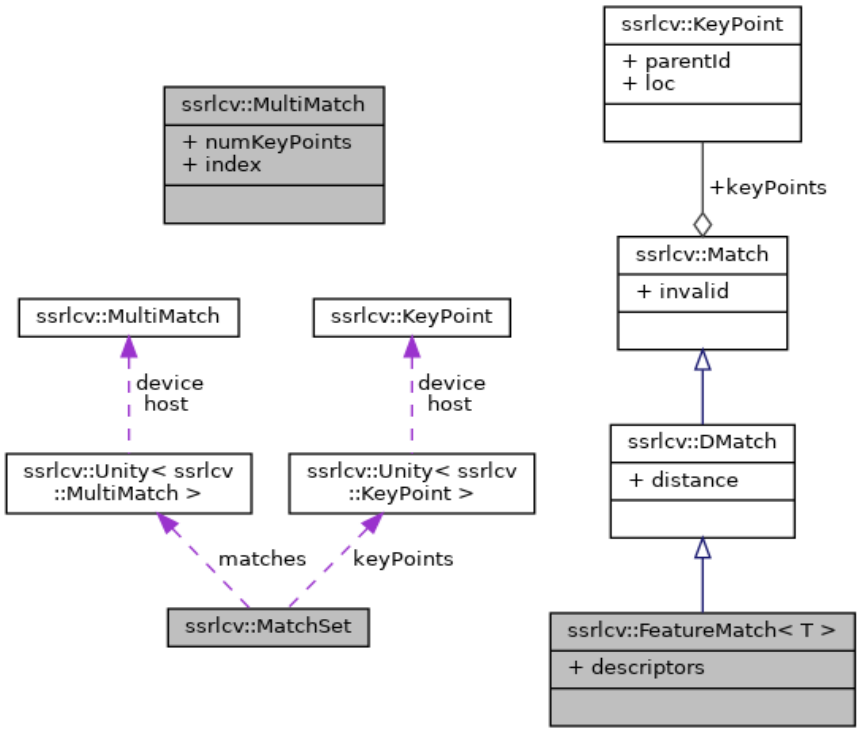

Figure 21 - Match UML Class Graph (right), MultiMatch UML Diagram (top left), MatchSet Class Graph (bottom left)

can be seen in figure 20. Moving forward, all that users need to do when implementing new feature detectors is ensure that the generated descriptor has a distProtocol method implemented, like the descriptors seen in figure 19, for compatibility with MatchFactory.

The MatchFactory class generates Match structures by taking in two or more Unity structures containing Feature arrays. All in all, this is a pretty simple class, but acts as a bottleneck for any pipeline utilizing many large features, like dense non-scale invariant SIFT. If there are more than two input images, then the system will generate a MatchSet that holds MultiMatch structures, which helps avoid forcing all matches to have the same number of key points. UML and class graphs of the structures produced by MatchFactory can be seen in figure 21. As stated above, this class will match any set of features where the contained Descriptor type has the distProtocol method implemented, which simply returns a floating-point value associated with the difference between that instance and

another instance of the same type. Other than exhaustive and brute-force matching, which is sufficient when feature set sizes are reasonable and GPUs are utilized, there are a few more intelligent options to choose from. These options include constrained matching, absolute thresholding and seed/relative thresholding. The way that constrained matching works is based on epipolar geometry derived from the relationship between the camera parameters of two overlapping images. When the user provides a fundamental matrix outlining that relationship, a linear epipolar line equation can be derived for each pixel narrowing the search space to a certain distance around that line [12]. This primarily helps with speed, but also contributes to improved accuracy. Absolute thresholding is certainly the easiest of these “intelligent” matching methods and simply helps reduce matching error by using a floating-point value that represents the maximum value returned from distProtocol that would constitute a valid match. The use of a seed thresholding is another concept derived from David Lowe’s work on the SIFT paper. This threshold is a value between 0 and 1 representing the maximum ratio of distProtocol return values the closest possible match to the closest impossible match; the closest impossible match coming from matching a seed image that is guaranteed to have no overlap. Matching between more than two images currently matches all combinations of images and then discards matches that do not completely interpolate with the matches between other images.

The purpose of PointCloudFactory is to generate 3D points from a set of matches generated by MatchFactory. Up until now all structures and work mentioned have been done by me, except for camera parameter adjustment methods. PointCloudFactory represents one area of the library that I have only contributed in terms of defining data

structures, debugging and testing. The credit for most of the work in algorithmic development here must go to Caleb Adams [2]. Implemented methods for point cloud generation that have been implemented include reprojection and stereo disparity. Stereo disparity is an extremely simple method and the concept is the exact same as the method 3D depth can be derived by having two eyes. Reprojection is much more complicated and completely reliant on camera parameterization but tends to produce much more desirable results and can utilize matches from more than two views. As mentioned previously, due to the possibility that camera parameters won't completely line up with the actual images, a global adjustment and parameter optimization system needed to be implemented as well in this step. This is known as bundle adjustment and is a common step in SfM pipelines to rectify error and improve the resulting point cloud [5].

Lastly, the next factory to discuss is the MeshFactory, which takes in a point cloud and produces a triangulated mesh. The primary components of this factory currently include the data parallel Octree and the marching cubes algorithm. These are two pieces to the Poisson Surface Reconstruction algorithm that Alex Lin recently took over developing. This is a well-known algorithm for producing smooth meshes from nonuniform point clouds. It does this through a few complicated steps that generally start with the distribution of point normal information onto octree vertices through 3D gaussians and ending with an indicator function that tells marching cubes what octree edges cross the surface boundary [16]. The octree that facilitates much of the computation for this and greatly speeds it up was implemented with the guidance of research conducted at Zhejiang University in conjunction with Microsoft Research Asia specifically for data-parallel octrees meant for surface reconstruction [57]. After understanding the concept of

that data structure, it was very easy to implement a data-parallel quadtree for inclusion in SSRLCV. One important use of the octree is point normal determination, primarily to speed up neighboring point searches. Once neighboring points are determined, the centroid of the set is determined, then a covariance matrix is formed by subtracting the neighboring point coordinates from that of the centroid. Finally, an SVD using the CUDA toolkit cuSOLVER library is performed on that matrix, where the resulting third eigen vector is the point normal [29]. The implementation of the GPU accelerated marching cubes algorithm was implemented after gaining a general understanding from the original Seminal Graphics paper [24] and then writing the cube configuration look up tables by hand for use with the octree vertex and edge ordering system.

Other components of the library act as utility. For every factory and class mentioned above, use of GPU stream compaction and sorting methods are essential to the speed of the system. Thrust, a library of the CUDA toolkit, is used with custom predicates for this purpose. The CUDA occupancy API is also used as an important tool

to ensure that kernels implemented within SSRLCV request optimal resources and can be executed on all CUDA capable devices. Along with that the use of Doxygen commenting is extensive and enabled some powerful hierarchical documents to facilitate future collaboration. Other than that, there are quite a few utility functions like IO and CUDA built-in vector operators that are not being mentioned in this thesis but do have Doxygen documentation entries.

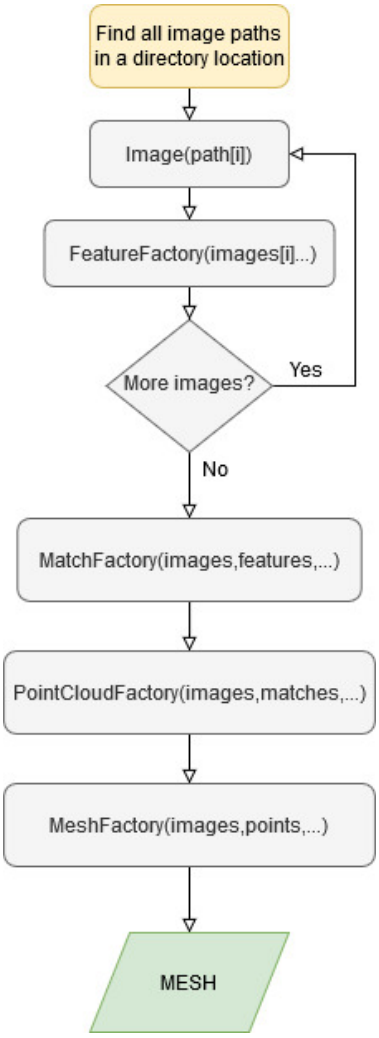

Figure 22 - SSRLCV Structure from Motion Implementation Flowchart

As stated before, this library was implemented to facilitate the execution of MOCI's SfM pipeline. By simply starting with image IO to fill multiple Image instances, the input data for the pipeline is ready. Each of those images can then be passed to a child FeatureFactory class, like SIFT_FeatureFactory, to generate feature sets. These feature sets are then passed to MatchFactory for match production. Once matches are available, they are passed to PointCloudFactory for 3D point cloud generation. After that MeshFactory will triangulate a mesh based on the point cloud. As can be seen in figure 22, the output of one factory is used as the input of the next.

This framework is being released open source on GitHub with a LGPL license, and providing the previously mentioned Doxygen documentation. Before release, some cleanup is occurring in terms of warning eradication, compilation optimization and further templating. Some of the useful lower level structures will also be released on GitHub as

mirrored repositories, like Unity as it is extremely useful on its own for CUDA programmers, especially beginners.

3.3.4 Neural Networks

Convolutional neural networks (CNN) are commonly used today for machine learning tasks. MOCI is utilizing two types of these networks, one proven to work for segmentation and the other to work for multi-object recognition and localization with bounding boxes. The purpose of the first and more important CNN being utilized on MOCI is for surface object recognition, directly related to mission objective MO-2. This model utilizes the YOLOv3 architecture, with the pretrained darknet weights for training, which has high degrees of success in recognition and labeling of multiple classes. This architecture, with the darknet backbone, was tested in the original paper on the 80 classes in the COCO dataset [22] including bow ties, cows, backpacks, etc. all with top-5 accuracy over 90% and an AP_{50} of 57.9 while being 3x faster than similarly accurate detectors [39]. Initial training was performed on the classes that had datasets with bounding box labels. This was to ensure that the network was functioning properly. The next step in training was to include images that had been directory labeled and cropped to only include the objects. The combined dataset is outlined in table 27 with the parent dataset and whether it was directory or bounding box labeled. The images acquired were from DOTA [54], Airbus Ship [3] and UCMerced [55], all having GSD specified. To make the models trained more accurate for the imagery MOCI will be acquiring, I scaled all images to have 7m GSD using the GPU accelerated SSRLCV bilinear interpolation scaling. During the next step I discarded any images that had bounding boxes smaller than 3 pixels wide after the down sampling. Finally, I added borders and resampled 512x512 so that every image had

the same dimension, which is known to be advantageous during the training process. Before training commenced, I also made sure to list at least 15% of the training sets for each class to act as the validation set used during training.

Table 27 - Object Recognition Classes and Training Dataset Details

Class Name	Dataset Details	Size
bridge	DOTA (labeled), UCMerced (directory)	250
sports_field	DOTA (labeled), UCMerced (directory)	250
plane	DOTA (labeled), UCMerced (directory)	250
harbor	DOTA (labeled), UCMerced (directory)	250
ship	Airbus Ship & DOTA (labeled), UCMerced (directory)	30,000
storage_tank	DOTA (labeled), UCMerced (directory)	250
mountain	UCMerced (directory)	100
river	UCMerced (directory)	100
wetland	UCMerced (directory)	100
seaorlake	UCMerced (directory)	100
island	UCMerced (directory)	100
forest	UCMerced (directory)	100
snoworice	UCMerced (directory)	100
beach	UCMerced (directory)	100
desert	UCMerced (directory)	100
agricultural	UCMerced (directory)	100
roadway	UCMerced (directory)	100
industrial	UCMerced (directory)	100
commercial	UCMerced (directory)	100
residential	UCMerced (directory)	100
stadium	UCMerced (directory)	100
golf_course	UCMerced (directory)	100
thermal_power	UCMerced (directory)	100

The next CNN, being used for cloud segmentation, did not need such strenuous dataset preparation or incremental training sessions. This CNN utilizes the UNET architecture which is well known for segmentation of and was initially released for neuron segmentation in brain MRIs [40]. Datasets used for training this model were from the visible light images of the 38-Cloud taken from Landsat 8 imagery dataset made up of images with clouds and masks [32]. The desired result of the trained network can be used for two purposes, one being cloud removal and the other being cloud focus. Cloud

removal is useful for focusing on ground reconstruction or narrowing the problem space of the object recognition task as that is only looking at ground objects. Cloud focus can facilitate another possible experiment on MOCI, cloud reconstruction and cloud height determination.

After validating the full functionality of the concept for both CNNs, the networks will be further trained on google cloud for an extended period to reach maximum accuracy. This stage could be called the “preparation for production” process and will utilize the confirmed network for pretrained weights. Once it is confirmed that the network cannot be improved anymore, the model weights will then be converted from their PyTorch form to an onnx format which is necessary in order to use TensorRT for optimization and minimization.

CHAPTER 4: RESULTS DISCUSSION

4.1 Presentation of testing/evaluation

To test the various portions of SSRLCV, we used of visualization scripts like our match plotter to show that we do indeed have a functional algorithm. As SIFT features are generally difficult to visualize in their final form, the method of verifying our CUDA accelerated features was through a rotation and check method. To fully test the rotation and scale invariance concepts that are inherent in the SIFT scale space implementation, images were rotated and blurred then matched. Figure 23 shows the individual tests for rotation invariance and scale invariance. The combined rotation and scale invariant tests can be seen in figure 24. Multiview match functionality was also confirmed in a similar way, which can be seen in figure 25.

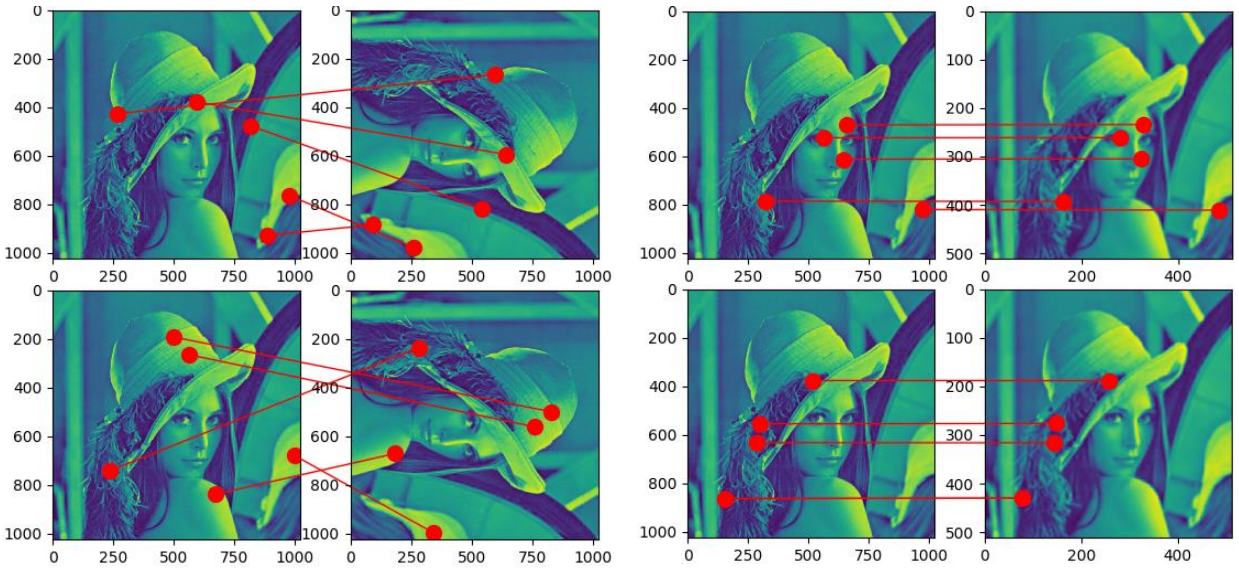

Figure 23 - SIFT Rotation Invariance (left) and SIFT Scale Invariance (right) Tests

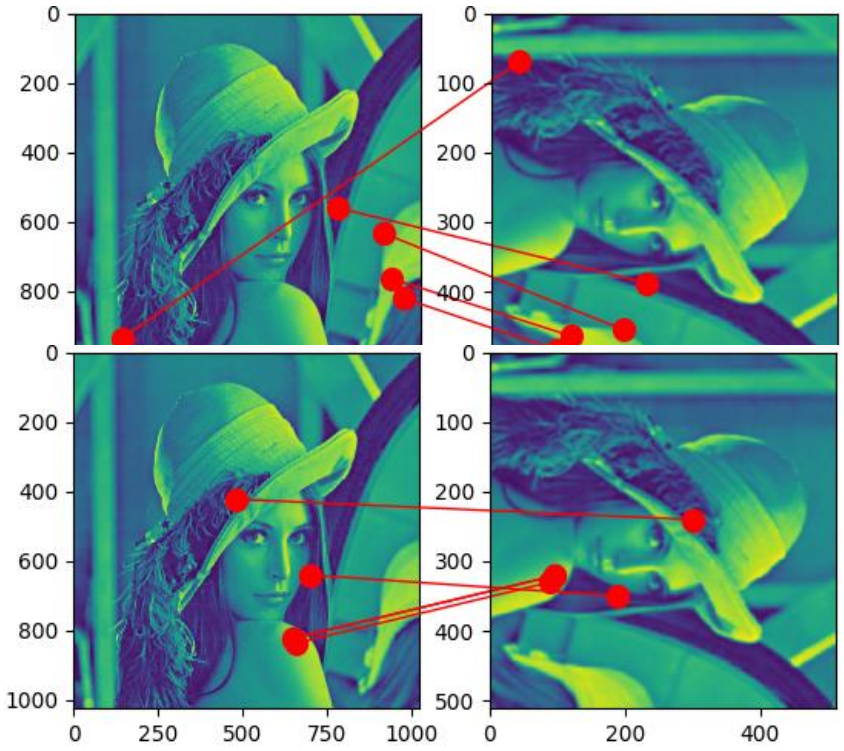

Figure 24 - SIFT Scale and Rotation Invariance Test

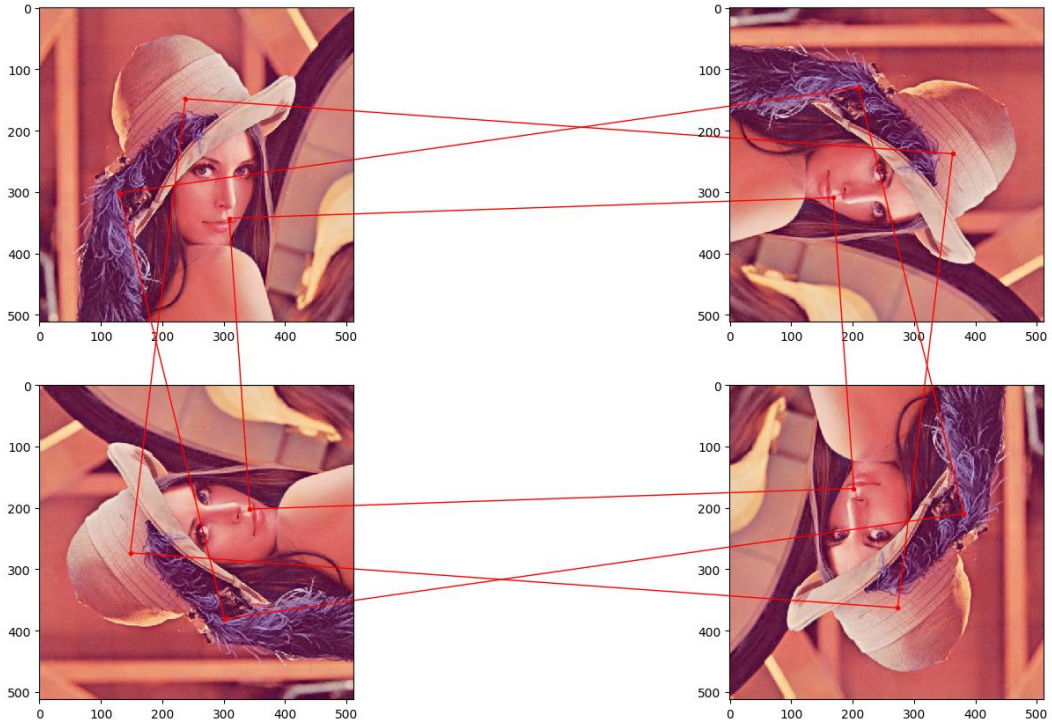

Figure 25 - N-view Matching with SIFT Features Test

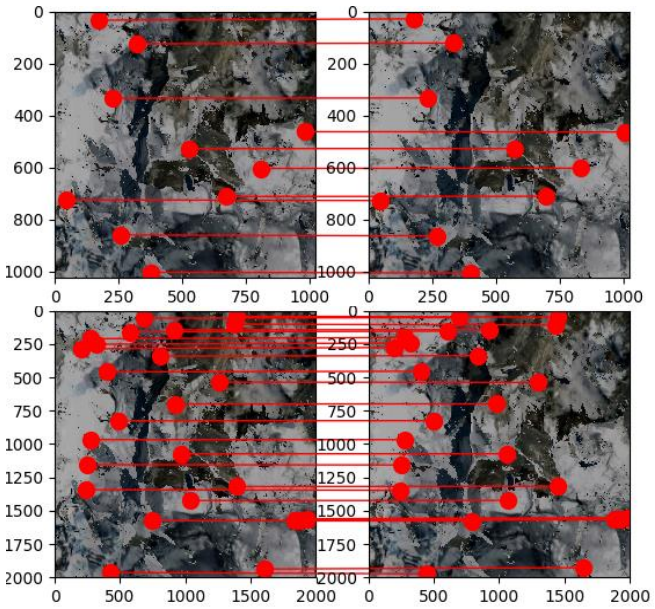

Figure 26 - Matches Utilized for Stereo Disparity

Now to test point cloud generation. Using the blender package that Jake Conley and Caleb Adams developed, we can test this in a realistic scenario modeling Mount Everest. Before reprojection was ready for testing, we tested stereo disparity to ensure that matches and viewpoint differences could generate point clouds with discernable elevation changes. Matches used in this initial test can be seen in figure 26 and the results of stereo disparity can be seen in figure 27. For reprojection testing we utilized 2, 3 and 5 view matches to generate the point clouds. These results can be seen in figure 28 a-c.

Figure 27 - SSRLCV Stereo Disparity Results

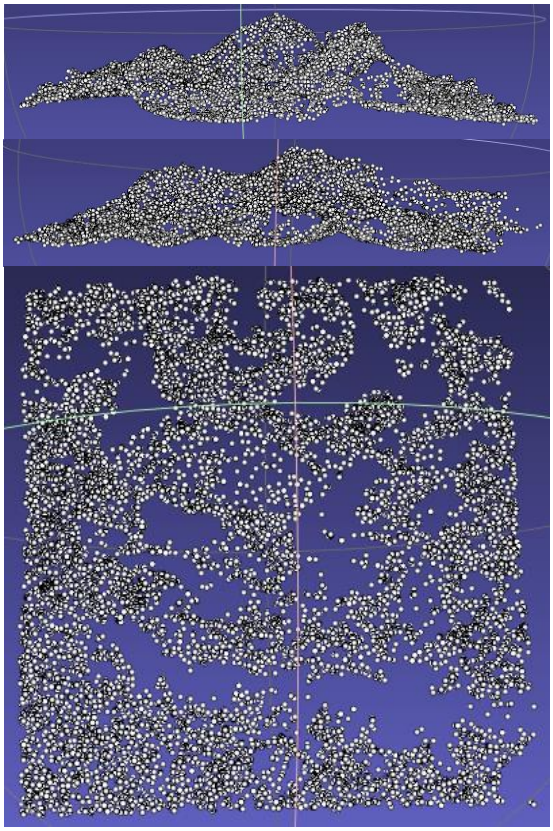

Figure 28 a) - 2-view Reprojection Results

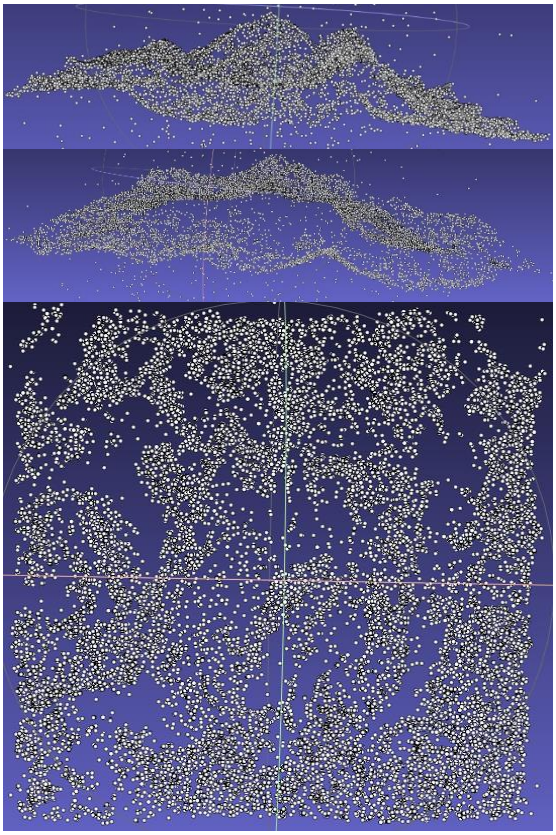

Figure 28 b) - 3-vew Reprojection Results

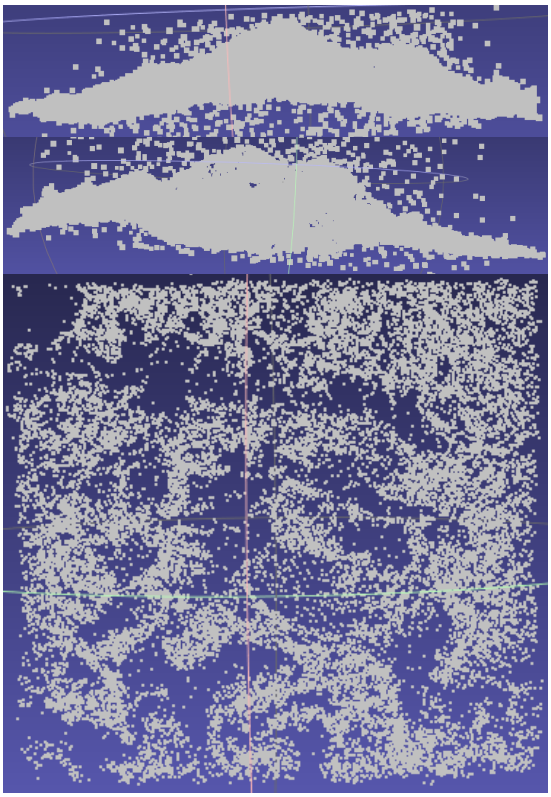

Figure 28 c) - 5-view Reprojection Results

As can be seen, the density of the cloud increases with multiple views, but so does the noise. Soon the 3- and 5-view point clouds will be as clean as the 2-view with the finalization of multiview bundle adjustment.

Other tests that were necessary to conduct mostly involved the Octree. We first tested point normal determination utilizing a point cloud from the well-known Utah Teapot. With that confirmed to work, we then tested an example of Everest. Results of these can be seen in figure 29. Next, to test the functionality of the marching cubes algorithm. Due to the necessity of an indicator function to compute an isosurface with marching cubes, the assumption that an octree edge crossed the surface was made if the closest point of its vertices has normals pointing in opposite directions. This was confirmed to at least show that marching cubes was functional, as seen in figure 30.

To test the functionality of neural networks after the first training phase, we utilized images from the validation set or the directory labeled UCMerced test set. For cloud segmentation, images were randomly selected from the validation set in 38-cloud

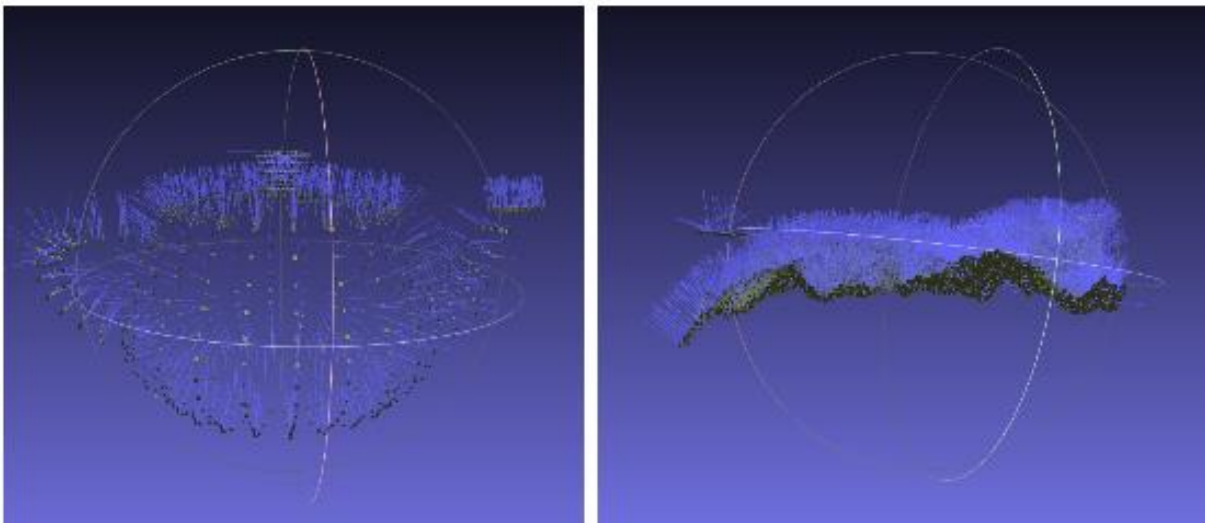

Figure 29 - SSRLCV Point Normal Approximation Results

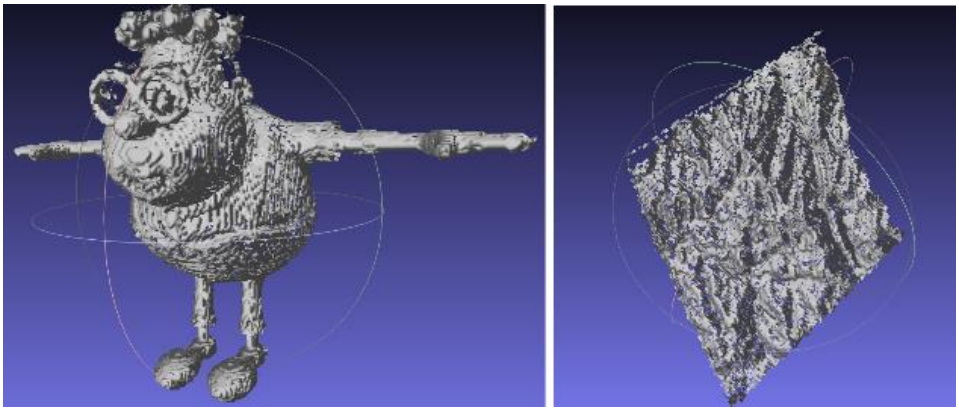

Figure 30 - SSRLCV Unsmoothed Marching Cubes Results

and ran through the segmentation program that utilized the trained weights. Results for this, as seen in figure 31, were fantastic and showing that the current model is very close to being accurate enough for conversion to TensorRT. After the first phase of training for object recognition promising results were seen for plane detection as well as the sports field category, seen in figure 32 and 33. This was not the case for ships, bridges, and storage tanks; likely caused by images used during training that had

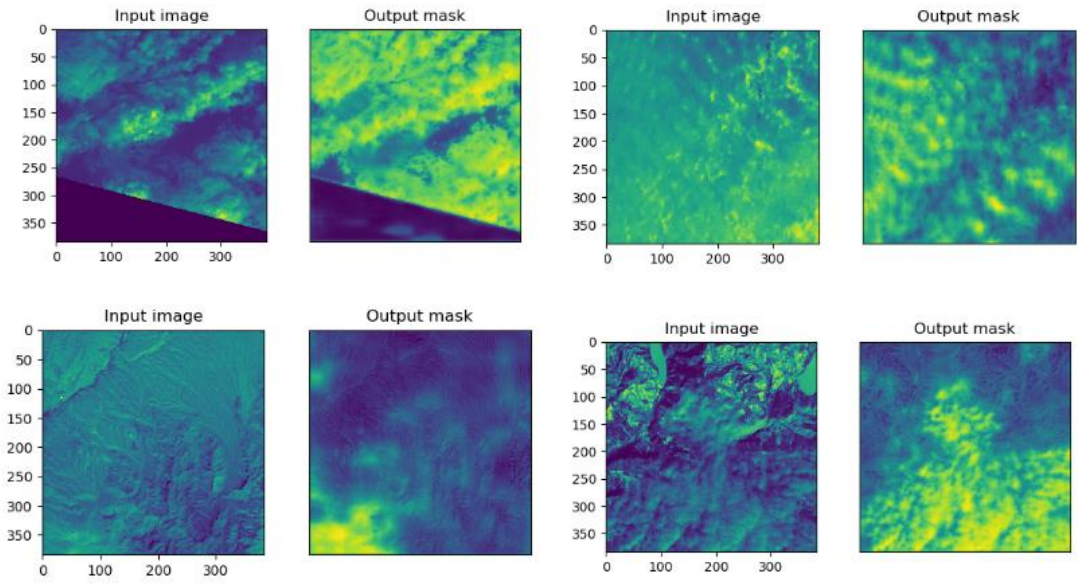

Figure 31 - UNET Cloud Segmentation Results

Figure 32 - Results of YOLOV3 plane Recognition After First Training Session

Figure 33 - Results of YOLOV3 Sports Field Recognition After First Training Session

objects binned so small that they were not recognizable. Due to this, there needs to be another phase of training after going through all the images by hand before directory

labeled training can be attempted.

4.2 Discussion of how results support overall MOCI mission objectives

Through careful planning, MOCI's payload software framework has grown to be an extensive and robust collection of individual components. The results displayed in section 4.1 show that the utilization of the SSRLCV framework and its current components can generate data products necessary to accomplish MOCI's first and second mission objective. With the finalization of Caleb Adam's work in N-view reprojection and bundle adjustment, MOCI will have all the software it needs for full mission success. Also, by building this framework with collaboration in mind, there should be little to no overhead for new students to develop and refine MOCI's payload software without having to reinvent anything.

After initial validation of the trained CNN architectures for cloud segmentation and object recognition, the results turned out to prove that the work done on this can be used directly on MOCI to fulfill its second mission objective. Though the development of the neural networks differed from the concept of developing a framework, the idea of passing on the projects was kept in mind, especially with all the work done in preparing datasets. Moving forward, the only thing that these networks can do is improve with utilization of things like google cloud resources and TensorRT for optimization.

The next mission objective, even though it relates to imagery, it does not directly relate to onboard processing, so results from this section would not directly benefit the objective as is. Success of the object recognition software however does provide a unique opportunity for automating the third mission objective. By adding a set of labeled

coastlines to the dataset and adding a class to the model definition, MOCI could use this to recognize coastline within the images FOV and downlink those images. Even if these events were scheduled, this could be beneficial to possibly reduce transmission of unnecessary data if there is any doubt that all images contain coastline. The last mission objective relates to training students and community outreach, so the results in this section do not have a huge impact on that. The rest of the work described in this thesis does. As systems engineer, I have spoken at community gatherings, like the Athens Amateur Radio club. I have also taken part in outreach events and mentored any student that is interested in payload engineering/payload software engineering.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Recap of outcomes of this work

The work of close to 75 individuals has led to MOCI reaching its current state in pre-integration. The groundwork laid before I took on the role of systems engineer allowed me to push the project towards the vision that manifested more than four years ago. After overseeing final design decisions and leading the process of hardware acquisition, all MOCI is waiting on is the delivery of hardware. Barring any further delays from COVID-19 preventing in person testing, MOCI should launch sometime mid-2021. Even though many of those in leadership positions mentioned throughout this thesis are leaving this May, the individuals that will carry the torch from here on out were chosen because their abilities have proven to be more than sufficient to ensure the MOCI mission is a success.

As the software engineer that designed MOCI's payload software framework, I am confident that those interested in further improving the system in the last year of MOCI's pre-launch lifetime will be able to jump in with ease. The system works as is and will be further supplemented by the work that is done in wrapping up the portions that are currently under development, mentioned throughout this thesis.

The success of this mission would not only validate the efforts here at the University of Georgia, but also put the SSRL on the map as a program capable of producing highly capable technology demonstrations that push the bounds of earth observation and on-orbit processing/intelligent operation.

5.2 Description of plans for MOCI mission future

Even though most of the design work for MOCI has been completed, a lot of work is still ahead of the mission. The next steps for MOCI are to get through the last two pass-fail reviews with integration occurring in between. The next review being PIR, which will take place this summer, is proposed to prove the team's capability of producing a functional satellite, in terms of electronics. Due to holding multiple internal PIR's, by the time UNP comes to evaluate in person, a pass should be a guarantee. By this time, it is planned that the mechanical team will have had a long head start on preparing for integration so that it can start immediately as we get the go ahead from UNP following the review. Throughout integration testing will occur to ensure that flight hardware is interfacing properly. Once the entire stack is together and MOCI is in its final form, the SSRL will repeat what was done on the flatsat during PIR to fully test the functionality of the flight unit. With confirmation of this, MOCI will hold PSR with UNP and hand off to them beginning Phase C of UNP in which environmental testing occurs and the satellite is in the AFRL's hands.

The future of MOCI's payload software framework includes the planned open source release accompanied with publishing a journal article on the library structure and the pipeline. As it will be released with the purpose of providing a comprehensive CUDA accelerated computer vision library, not just for execution on an NVIDIA Jetson TX2i, further benchmarking will be conducted on a variety of devices. Comparison to similar pipelines developed in OpenCV will also be conducted for that publication as not all situations necessitate a robust and light weight solution like something running on a satellite. Along with the exciting future that MOCI's payload software has, the hardware

recently was offered and excellent opportunity. Due to some overlap with the payload being utilized with on the University of Texas SERPENT mission, CORGI is being utilized and tested on a weather balloon and potentially a spaceshot rocket meant to get to the Von Karman line. We are also being provided with a chance to run our own payload software in an environment like what MOCI will see, with very little cost to us. A weather balloon test will also be included in this project for more extreme environment testing of the hardware. The target is looking like a weather balloon test sometime this summer and a high-altitude launch test by the end of the year.

5.3 Potential spin-off opportunities from this project

A few major spin-off opportunities come to mind when considering the research that MOCI is conducting. First, a second iteration of MOCI would lead to meaningful improvements. With a usable funding pool about 600k USD, an improved ADCS system could seriously improve the tracking capability of MOCI while the rest of the non-payload stack could stay the same. A great choice for this would be the Blue Canyon XACT, which was unfortunately not an option for MOCI as the trade study forced to be reopened very late in the development process. To improve camera telemetry association, adding a direct line of communication between the payload processor and ADCS would be great benefit as well. Any improvement on the optical train to gain a lower GSD would also be desired. Iterating on this design and improving the systems performance and sustainability in extreme radiation rich environments would provide a wide range of interesting applications. For instance, deployment for analysis and monitoring of other planetary bodies. The capability to process high resolution images on board would minimize the data transmission requirements, which could be costly if orbiting something

like one of Jupiter's moons. Doing this on a low-cost spacecraft would allow for repeatability and further coverage, while also allowing experimentation and risk to persist without being too much of a danger to cost effectiveness. By utilizing that concept of repeatability, another opportunity presents itself, small satellite swarms or constellations. A constellation of satellites like MOCI could be utilized to facilitate a high coverage near-real-time monitoring or surveillance system for a planetary body.

References

[1] *2014 Nano/Microsatellite Market Assessment". annual market assessment series. Atlanta, Georgia: SEI. January 2014: 18. Archived from the original on 22 February 2014. Retrieved 18 February 2014.*

[2] Adams, C. A., (2020). *High Performance Computation with Small Satellites and Small Satellite Swarms for 3D Reconstruction* [Unpublished master's thesis]. University of Georgia

[3] Airbus Ship Detection Challenge. (n.d.). Retrieved from https://www.kaggle.com/c/airbus-ship-detection/data

[4] Berthoud, L., Swartwout, M., Cutler, J., Klumpar, D., Larsen, J. A., & Nielsen, J. D. (2019). University CubeSat Project Management for Success. Retrieved from https://digitalcommons.usu.edu/smallsat/2019/all2019/63/

[5] Choudhary, S., Gupta, S., & Narayanan, P. J. (2012). Practical Time Bundle Adjustment for 3D Reconstruction on the GPU. *Trends and Topics in Computer Vision Lecture Notes in Computer Science*, 423–435. doi: 10.1007/978-3-642-35740-4_33

[6] Clements, J., Murphy, T., Jasper, L., & Jacka, C. (2019). Tailored Systems Engineering Processes for Low Cost High Risk Missions. Retrieved from https://digitalcommons.usu.edu/smallsat/2019/all2019/61/

[7] Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observations (CALIPSO). (2007). *Van Nostrands Scientific Encyclopedia*. doi: 10.1002/9780471743989.vse9856

[8] Cotten, D., Adams, C., & Neel, N. (2017). The Feasibility of Structure from Motion Over Planetary Bodies Using Small Satellites. Retrieved from https://digitalcommons.usu.edu/smallsat/2017/all2017/37/

[9] Cotten, D., Neel, N., et al. (2018). The Spectral Ocean Color (SPOC) Small Satellite Mission: Developing an Adjustable Multispectral Imager. In *AGU Fall Meeting Abstracts* (pp. ED43G-1295).

[10] CubeSpace. (2019, September 27). CubeADCS User Manual [User Manual]

[11] CUDA Toolkit Documentation v10.2.89. (n.d.). Retrieved from https://docs.nvidia.com/cuda/

[12] Epipolar Geometry and the Fundamental Matrix. (2004). *Multiple View Geometry in Computer Vision*, 239–261. doi: 10.1017/cbo9780511811685.014

[13] FAQ Full General Public Webb Telescope/NASA. (n.d.). Retrieved from https://www.jwst.nasa.gov/content/about/faqs/faq.html#partners

[14] Gardner, J. P. (2007). James Webb Space Telescope (JWST). *Van Nostrands Scientific Encyclopedia*. doi: 10.1002/9780471743989.vse9979

[15] HajiRassouliha, A., Taberner, A. J., Nash, M. P., & Nielsen, P. M. F. (2018, July 25). Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. Retrieved from https://www.sciencedirect.com/science/article/pii/S0923596518303606

[16] Hoppe, H. (2008). Poisson surface reconstruction and its applications. *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling - SPM 08*. doi: 10.1145/1364901.1364904

[17] It's not an embedded Linux distribution – it creates a custom one for you. (n.d.). Retrieved from https://www.yoctoproject.org/

[18] Jetson Download Center. (2020, March 5). Retrieved from https://developer.nvidia.com/embedded/downloads#?search=Data Sheet

[19] Johnson, N. L., Hoffman, E., Forsgren, R., Dodge, S., Bonilla, E., & Srey, T. Orbital Debris Managment & Risk Mitigation. (D. Connell, M. Kohut, H. Stephenson, & D. Connell, Eds.), Orbital Debris Managment & Risk Mitigation (n.d.).

[20] K, S. (2013). Lossless LZW Data Compression Algorithm on CUDA. *IOSR Journal of Computer Engineering*, *13*(1), 122–127. doi: 10.9790/0661-131122127

[21] Lee, J.-W., Kim, B., & Yoon, K.-S. (2014). CUDA-based JPEG2000 encoding scheme. *16th International Conference on Advanced Communication Technology*. doi: 10.1109/icact.2014.6779047

[22] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, 740–755. doi: 10.1007/978-3-319-10602-1_48

[23] Loff, S. (Ed.). (2015, July 20). CubeSats. Retrieved from https://www.nasa.gov/mission_pages/cubesats/index.html

[24] Lorensen, W. E., & Cline, H. E. (1998). Marching cubes. *Seminal Graphics*, 347-353. doi: 10.1145/280811.281026

[25] Lowe, David G. “Distinctive Image Features from Scale-Invariant Keypoints.” *International Journal of Computer Vision*, vol. 60, no. 2, 2004, pp. 91–110., doi:10.1023/b:visi.0000029664.99615.94.

[26] Lynk Joins the Satellite Industry Association. (n.d.). Retrieved from https://2019.smallsatshow.com/2019/09/20/lynk-joins-the-satellite-industry-association/

[27] Martines, Luz Maria S., "Analysis of LEO Radiation Environment and its Effects on Spacecraft's Critical Electronic Devices" (2011). Dissertations and Theses. 102. https://commons.erau.edu/edt/10

[28] Maurer, J. (2001). Overview of NASA's Terra satellite. Retrieved from https://www2.hawaii.edu/~jmaurer/terra/

[29] Mitra, N. J., & Nguyen, A. (2003). Estimating surface normals in noisy point cloud data. *Proceedings of the Nineteenth Conference on Computational Geometry - SCG 03*. doi: 10.1145/777792.777840

[30] Mlambo, R., Woodhouse, I., Gerard, F., & Anderson, K. (2017). Structure from Motion (SfM) Photogrammetry with Drone Data: A Low Cost Method for Monitoring Greenhouse Gas Emissions from Forests in Developing Countries. *Forests*, *8*(3), 68. doi: 10.3390/f8030068

[31] MLI: Satkit. (n.d.). Retrieved from https://www.dunmore.com/products/satkit-mli.html

[32] Mohajerani, S., Krammer, T. A., & Saeedi, P. (2018). A Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks. *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. doi: 10.1109/mmsp.2018.8547095

[33] Moog Inc. (n.d.). Orbital Maneuvering Vehicle. Retrieved from https://www.moog.com/markets/space/omv.html

[34] NanoRacks DoubleWide Deployer System Interface Definition Document RevA Retrieved from https://nanoracks.com/wp-content/uploads/NanoRacks-DoubleWide-Deployer-NRDD-Interface-Definition-Document.pdf

[35] NASA. 2012. "Process for Limiting Orbital Debris." standards.nasa.gov.

[36] Orbital Debris Mitigation Standard Practices, November 2019 Update, Orbital Debris Mitigation Standard Practices, November 2019 Update (n.d.).

[37] Patyuchenko, A., Younis, M., & Krieger, G. (2015). Compact X/Ka-band dual-polarization spaceborne digital beamforming Synthetic Aperture Radar. 2015 16th International Radar Symposium (IRS). doi: 10.1109/irs.2015.7226335

[38] PRISMA: small Innovative Earth Observation mission. (n.d.). Retrieved from http://prisma-i.it/index.php/en/

[39] Redmon, J. (2018, April 8). YOLOv3: An Incremental Improvement. Retrieved from https://arxiv.org/abs/1804.02767

[40] Ronneberger, O. (2017). Invited Talk: U-Net Convolutional Networks for Biomedical Image Segmentation. *Informatik Aktuell Bildverarbeitung Für Die Medizin 2017*, 3–3. doi: 10.1007/978-3-662-54345-0_3

[41] Satellite Imagery and Archive. (2019, April 29). Retrieved from https://www.planet.com/products/planet-imagery/

[42] *"Technology Readiness Level Definitions" (PDF). nasa.gov. Retrieved 6 September 2019.*

[43] Todd, David. 2017. "Final Score for 2017: 466 – a New Record for the Number of Satellites Attempted to be Launched in a Single Year" (Corrected). December, 2017. https://www.seradata.com/final-score-for-2017-463-a-new-record-for-the-number-of-satellites-attempted-to-be-launched-in-a-single-year/

[44] Thome, K. (n.d.). About Terra. Retrieved from https://terra.nasa.gov/about

[45] Transon, J., D'Andrimont, R., Maugnard, A., & Defourny, P. (2018). Survey of Hyperspectral Earth Observation Applications from Space in the Sentinel-2 Context. *Remote Sensing*, *10*(3), 157. doi: 10.3390/rs10020157

[46] Tummala, A. R., & Dutta, A. (2017). An Overview of Cube-Satellite Propulsion Technologies and Trends. *Aerospace*, *4*(4), 58. doi: 10.3390/aerospace4040058

[47] Tyvak Missions. (n.d.). Retrieved from https://www.tyvak.com/missions/

[48] University of Georgia Small Satellite Research Lab. (2019, November 03). MOCI Electrical Ground Support Equipment Design [Analysis]

[49] University of Georgia Small Satellite Research Lab. (2020, February 22). Power Budget and Power Budget Engine Overview [Analysis]

[50] University of Georgia Small Satellite Research Lab. (2019, November 03). Radiation Mitigation Design [Analysis]

[51] University of Georgia Small Satellite Research Lab. (2020, February 20). Thermal Analysis [Analysis]

[52] UNP Missions. (n.d.). Retrieved from https://universitynanosat.org/missions/

[53] Varghese, K. S., Pandey, M. C., Radhakrishna, K., & Bawa, A. S. (2014). Technology, applications and modelling of ohmic heating: a review. *Journal of food science and technology*, *51*(10), 2304–2317. https://doi.org/10.1007/s13197-012-0710-3

[54] Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., … Zhang, L. (2018). DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2018.00418

[55] Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.

[56] Yost, B. (2019, January 7). Executive Summaries. Retrieved from https://sst-soa.arc.nasa.gov/executive-summary

[57] Zhou, K., Gong, M., Huang, X., & Guo, B. (2011). Data-Parallel Octrees for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, *17*(5), 669–681. doi: 10.1109/tvcg.2010.75

[58] Zwally, J. (2003). GLAS/ICESat L1B Global Elevation Data. *National Snow and Ice Data Center Datasets*. doi: 10.3334/nsidc/gla01

Appendix

Table 28 - Examples of Public Small Satellites

Organization	Mission/Design	Size	Details	Launch
Planet Labs [41]	PlanetScope	CubeSat	Earth Imaging @ 3m GSD	>130x
	SkySat	CubeSat	Earth Imaging @ 5m GSD	15x
	RapidEye	CubeSat	Earth Imaging @ 72cm GSD	5x
TYVAK [47]	NANOACE	3U CubeSat	Rendezvous and Proximity Operations	
	CICERO	6U CubeSat	Climate and Weather	
	GEOSTARE	3U CubeSat	Situational Awareness	
	RAINCUBE	6U CubeSat	Tropical Weather Prediction	
	PATHFINDER	6U CubeSat	Multiple Payloads	
	LunIR	6U CubeSat	IR for Lunar Surface	
	RUNNER	CubeSat	Earth Imaging @ 71cm GSD	
	CPOD	3U CubeSat	Proximity Operations	
	SAR	6U CubeSat	Synthetic Aperature	
NASA [23]	CAPSTONE	12U CubeSat	Autonomous Positioning	Planned 2021
	Lunar Flashlight	CubeSat	Lunar Mapping	Planned 2021
	HARP	3U CubeSat	Hyper-Angular Rainbow Polarimeter	2019
	CIRiS	6U CubeSat	Compact Infrared Radiometer in Space	2016
	MarCO	6U CubeSat	Mars Cube One	2018
	BioSentinel	6U CubeSat	Hyper-Angular Rainbow Polarimeter	Planned 2020
UNP [52]	3-CORNERSAT	Microsatellite	Stereoscopic Imaging, Distributed	2004

			Operations, and Virtual Formation Flying Operations and Communications	
	Ho`oponopono	3U CubeSat	Orbital Radar Calibration	2013
	COPPER	3U CubeSat	compact uncooled microbolometer array	2013
	Argus	2U CubeSat	modeling of radiation effects	2015
	CHOMPTT	3U CubeSat	enhanced GPS	2018
	PolarCube	3U CubeSat	tropospheric temperature sounding	2019
	Armadillo	3U CubeSat	Space debris measurement	2019
	MSAT	Microsatellite	Proximity Operation Testing	Planned 2020+
	RECONSO	6U Cubesat	Space-based Surveillance	Planned 2020+
	GLADOS	6U CubeSat	collect multi-band photometric data	Planned 2020+
	MAXWELL	6U CubeSat	high speed and resilient advanced communication system	Planned 2020+

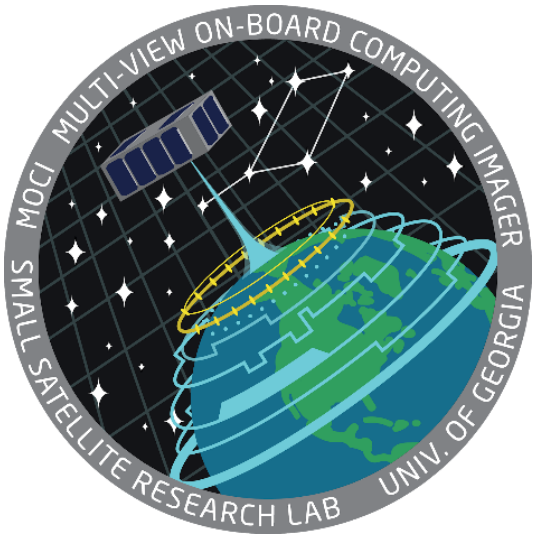

Figure 34 - MOCI Logo

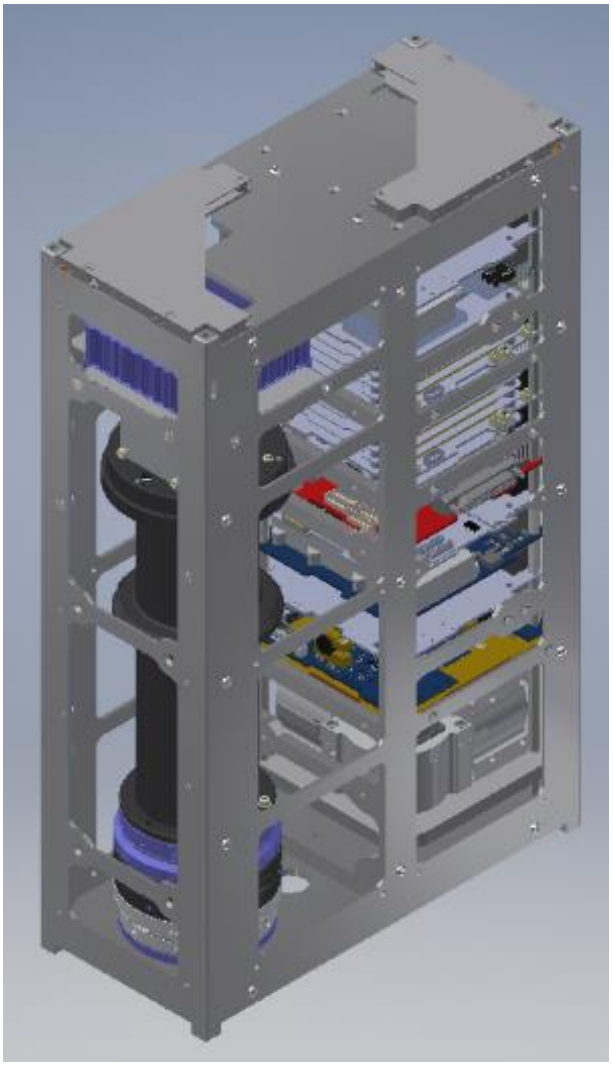

Figure 35 - MOCI's Full Stack

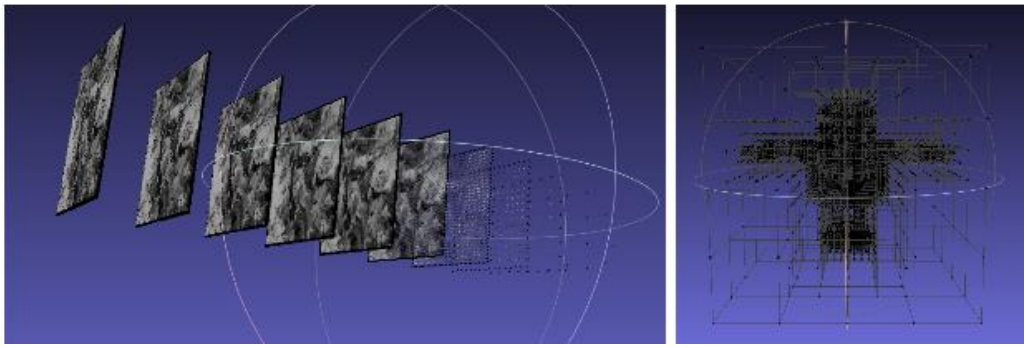

Figure 36 - SSRLCV Tree Examples