University of Colorado
Department of Aerospace Engineering Sciences
Senior Projects - ASEN 4018
Kinesthetic Engineered Solution to Space Litter and Exhausted Resources
Conceptual Design Document

October 3, 2017

I. Information

A. Project Customer

NAME: Josh Stamps
ADDRESS: 1722 Boxelder St, Louisville, CO 80027
Email: josh.stamps@sncorp.com
Phone: 720-407-3178

B. Group Members

Name: Glenda Alvarenga Email: glal4280@colorado.edu Phone: 720-277-6053	Name: Abdiel Agramonte-Moreno Email: abag5006@colorado.edu Phone: 702-321-4546
Name: Thanh Cong Bui Email: thanh.bui@colorado.edu Phone: 303-547-6142	Name: Christopher Choate Email: Christopher.Choate@colorado.edu Phone: 509-951-3118
Name: Lauren Darling Email: Lauren.Darling@colorado.edu Phone: 817-676-6154	Name: Sergey Derevyanko Email: sede1371@colorado.edu Phone: 720-300-1107
Name: Cassidy Hawthorne Email: Cassidy.hawthorne@colorado.edu Phone: 330-242-1411	Name: Abigail Johnson Email: abjo5441@colorado.edu Phone: 720-883-6608
Name: Nick Thurmes Email: nith2593@colorado.edu Phone: 720-600-4666	Name: Jannine Vela Email: jave9878@colorado.edu Phone: 720-448-4416
Name: Taylor Way Email: taylor.way@colorado.edu Phone: 720-290-8211	

Contents

II. Project Description

A. Project Overview

In recent years the Low Earth Orbit (LEO) environment has been subjected to an increase in orbital debris, which is predicted to triple by 2030[1]. This percentage is continuously increasing due to various reasons, including: pieces of satellite components that have broken off or become detached, satellites without propulsion at the end of their mission life, disabled/malfunctioning satellites, launch vehicle upper stages, and other miscellaneous man-made space junk.

Sierra Nevada Corporation (SNC), in collaboration with the Ann H.J Smead Aerospace Engineering Department undergraduate senior design team at the University of Colorado at Boulder, seeks to address this issue by developing technologies that demonstrate the capability to remove space debris using a low-cost small satellite with a robotic arm. The concept of this robotic arm satellite has three operational objectives: (1) perform rendezvous with the space debris, (2) grasp the space debris with the robotic arm, and (3) use one of several methods to de-orbit the space debris or move the debris to a different orbit from lower Earth orbit (LEO).

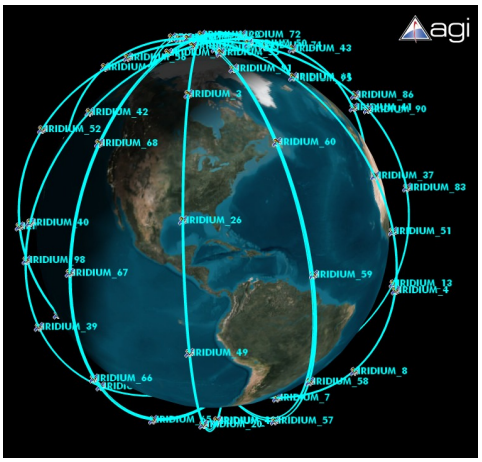

Figure 1: The Iridium constellation 2009 (AGI) demonstrating high prominence in LEO.[2]

The Kinesthetic Engineered Solution to Space Litter and Exhausted Resources (KESSLER) project will address the second operational objective, 'grappling space debris with a robotic arm'. The type of space debris that will be used to demonstrate this capability is the Iridium satellite constellation due to it's prominence in LEO and relevance to SNC's needs as seen in Figure 1 (further details will be discussed in the Ground and Test Support subsystem trade sections). A successful project will demonstrate: identification of the satellite and satellite grappling features (which are defined as: solar panel joints, bus support structure, and communication antennas) via image processing, determination of grappling approach paths to the grappling features (and collision avoidance), and capture of the grappling features via a multiple-degree-of-freedom (DOF) robotic arm.

This project is a follow-on effort from the 2016-2017 senior design project CASCADE, which demonstrated the capability to grapple a spinning satellite using a five degree-of-freedom Crustcrawler arm, using an external imaging facility to track the dynamics of the spinning satellite. KESSLER will not only improve the existing range of motion of the arm, but it will also remove the need of using an external facility to characterize the target object (and grappling location). By doing this, KESSLER will improve the the ability to integrate its system to a spacecraft that will be developed by SNC in the future. This project will be conducted in a controlled lab space at the University of Colorado (1g, non-vacuum environment, with sufficient lighting) and will not require space-qualified hardware/standards.

B. Project Objectives

As discussed in Sec. II.A, the purpose of this project is to address the second operational objective of SNC, which is to grapple the space debris with a robotic arm. Three driving assumptions have been defined that will bound the function of this project, as seen below:

1. The target object is in front and within reach of the robotic arm; this entails that this scenario is valid if the target object and the chase vehicle (spacecraft with arm) are in the same orbit and in proximity to each other.
2. The target object is stationary with respect to the robotic arm; this entails that the scenario is valid if the target object is 3-axis stabilized.
3. The target object and chase vehicle are illuminated, and the grappling procedure occurs during a Sun-Soak operation (about 45 minutes in low-Earth orbit); this entails that the scenario is valid if there is both sufficient lighting for Red Green Blue (RGB) sensors to resolve edges of the target object and if the grappling procedure duration is less than 45 minutes.

An additional assumption that has been made in order to demonstrate the project objectives is to scale down the size of the Iridium satellite model. This size would be determined by considering the capabilities of the the existing robotic arm and also considering potential testing facility resource limitations. This was discussed with and approved by the customer, since the image processing algorithm will still be applicable with a larger scale satellite and the hardware that can support it (i.e. larger robotic arm and compatible control sensors). Based on these assumptions (project entry criteria), the primary objectives of the project are to:

1. Take visual data confirming the target object (satellite) is in KESSLER's Field of View (FOV)
2. Identify pre-defined grappling features (PGFs) on the target object with an unknown orientation
3. Determine a prediction path to the target while avoiding collisions.
4. Autonomously grapple the PGFs on the target object via the robotic arm.

SNC has asked for specific objectives, as defined below:

- Modification of the existing robotic arm to include additional degrees of freedom necessary to grapple from a variety of approach angles.
- Incorporation of an RGB sensor into the existing machine vision system that currently uses only a LIDAR distance sensor.
- Incorporation of machine learning algorithm to enable autonomous recognition of the designated grappling feature.

The additional DOF has initially been chosen to be a rotation capability at the midway bend (will be discussed in the mechanical arm subsystem trade sections) - this will enable the arm to grapple objects that are not in the same plane as the base plate.

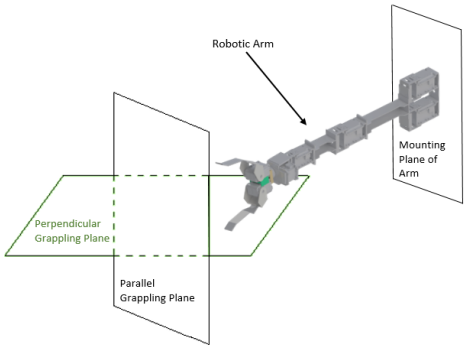

Figure 2: Grappling Plane Definitions

Based on the needs of the customer, the PGFs will be defined as physical structural items that protrude on the spacecraft that can be used to grapple. In particular these will be solar panel fixtures, bus support structure, and communication antennas on the Iridium satellite class. A structural mock-up of this spacecraft (although down scaled in size by a factor of approximately 65 $\%$ to simplify test resource needs) will be created by the KESSLER team.

Table 1 below defines the various levels of success for the major aspects of this project. Planes are defined to be with respect to the mounting of the arm versus the plane of the end-effector (see Figure 2 to view mounting with respect to the arm's end-effector; this is an example of them being parallel). Perpendicular planes would be any plane that intersects the normal plane (i.e. Figure 2 only shows one perpendicular case). The remaining orientations would be considered angled planes.

Table 1: Level Definitions

	Visual Processing		**Control**	**Control & Robotic Arm**
	Object Identification	**Processing**	**Path Prediction**	**Command Execution**
Level 1	Identify at least two surfaces on the satellite with varying depths in 3D space.	Identify the distance between the closest point of the satellite and the base of the robotic arm ($\pm$ 4mm).	Define travel path of robotic arm for end-effector to arrive at closest point on the satellite.	Demonstrate end-effector can move to closest point while facing the parallel plane.
Level 2	Identify grappling feature recognition on target satellite.	Determine grappling feature location and orientation to within $\pm$4mm & $\pm$ 5 deg.	Define travel path of robotic arm for end-effector to obtain as well as end-effector orientation required to arrive and grapple feature.	Grapple feature in parallel plane within $\pm$ 90 deg end-effector roll angle.
Level 3	Identify collision features on target satellite.	Define keep-out zone to within $\pm$ 4mm of collision feature surface, and select grappling feature of less collision risk.	Define constrained travel path of robotic arm for end-effector to obtain to arrive at grapple feature as well as end-effector orientation required.	Grapple feature in perpendicular plane (demonstrate additional ROM)

C. Project Functionality

In order to satisfy assumptions 1 and 2 as discussed in Section.II.B a Mechanical Ground Support Equipment (MGSE) fixture will be utilized (robotic arm and satellite structure will be mounted at a fixed distance). Once mounted on to the MGSE, the Crustcrawler arm will be supported on a platform (base plate) allowing the mobility for six degrees of freedom (DOF) in a 1G environment. Additionally a primary RGB camera will be mounted on that same

baseplate and will face the spacecraft model. The spacecraft model will also be mounted on the MGSE at a fixed distance away from the base plate of the Crustcrawler arm (distance is defined by the first functional requirement as seen later in this section). Power will be provided to both the central processing unit (CPU) and a peripheral electrical component (smaller secondary camera) mounted on the arm (motors within arm, and on claw respectively). One of the major lessons learned from the heritage project, CASCADE was that the servo motors may overheat and halt actuator motion after operating for 4 minutes - though this was not fully characterized the KESSLER team aims to prevent this by bounding the motor operation not exceed 4 minutes (this will require some characterization testing to identify the final metric) which will terminate actuation prior to allowing the motors to overheat and become damaged. The smaller secondary camera, will be mounted on the claw of the robotic arm and will be utilized for fine adjustments when the claw approaches a grappling feature.

A user will initiate the KESSLER algorithm via the CPU, this algorithm is comprised of: (1) an image processing segment and (2) software control segment. The software control segment will initiate once it has interpretted the feature location the image processing segment has determined based on the primary camera's image. After actuating to within a distance (which will be determined during feasibility studies) away from the feature, the secondary camera will take an image and this image data will be used to allow the software control segment to command the arm to make any fine adjustments prior to capture. The algorithm will command the motor drivers to actuate and rotate the mechanical arm to a nominal grappling position and close for capture and will use the encoders that are within the motors to confirm actuation. .

Data from the secondary camera will be collected directly by the CPU and the force sensors on the claw will relay information to the CPU via a myRIO DAQ and the algorithm will verify successful capture. A depiction of how the various segments and hardware module interact is shown in Figure 3.

1. Functional Requirements

Functional requirements are identified as follows:
F1. Robotic arm shall utilize visual processing system to locate target object in its line of sight to within a 30.5 inch (77.47 cm) linear distance from mounting base to the closest edge of said object. (*Length corresponds to maximum Extension of Arm, and visual processing are both customer constraints*)
F2. The robotic arm shall capture at least one pre-selected grappling feature on target object. (*Customer Constraint*)
F3. F3: Positioning algorithm shall provide a path for target location to capture target object.
F4. F4: The robotic arm shall autonomously grasp feature on target object.
F5. F5: The KESSLER system shall have a total process operations time defined by the Sun-Soak phase of an average LEO orbit (total mission phase 45 minutes $\pm$10 minutes).

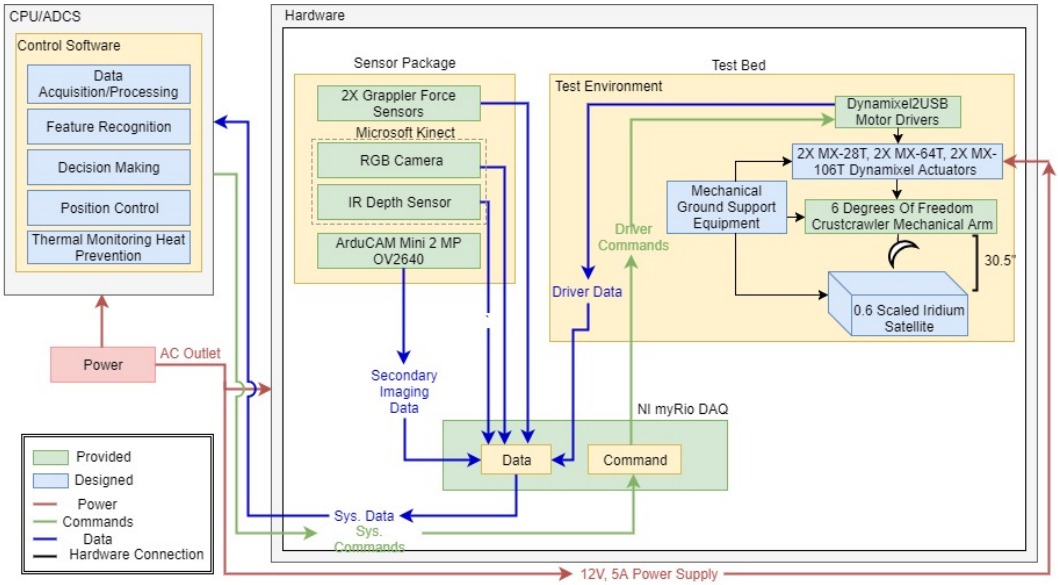

Figure 3: Functional Block Diagram

2. *Concept of Operations*

There are three steps needed in order to remove space debris from orbit: (1) perform rendezvous with the space debris, (2) grasp the space debris with the robotic arm, and (3) perform maneuver to de-orbit space debris. This general CONOPS is found in Figure 4.

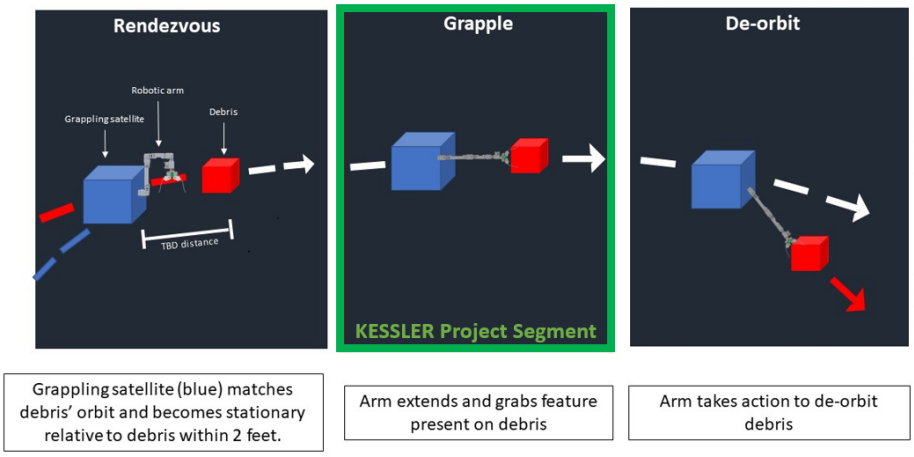

Figure 4: Mission Overview: KESSLER will address the Grappling Segment

For this project, only the second objective will be addressed. There are multiple scenarios that need to be taken into account. Since one specific feature will be grappled, and the orientation of the space debris is unknown, the arm must be able handle a case in which the feature is initially out of sight of the camera. Nominally, the primary camera will take an image and identify a feature, the arm will then maneuver near that location and use the secondary camera mounted on the claw to take a second image so that the control algorithm can command any minor of adjustments. Once 'aligned' with the feature, the claw will move toward the feature and grapple it. Force sensors will then send feedback to system to confirm capture.

In the event that the feature is on the part of the debris which faces the robotic arm, the arm will run a visual scan, identify the desired feature, decide the best path for the end-effector to get there, and then grapple the object at that feature. A software logic flow chart can be seen in the appendix.

If the feature is not seen on the first visual scan, a search algorithm is run which moves the end-effector around about 90 degrees to the right or left (depending on the shape of the debris). If the feature is found and it is accessible to the grappling arm it will be grabbed. If the feature is still not found or it is too hard to reach with the arm, the arm will suggest to be moved 90 degrees with respect to the satellite (depending on the location of the feature and shape of the debris). This is shown below as Figure 5.

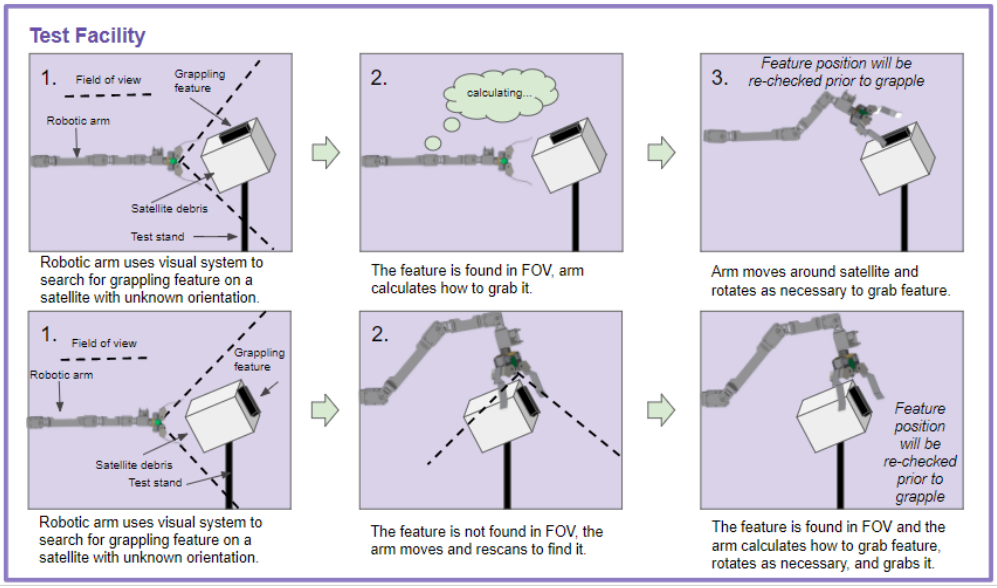

Figure 5: KESSLER Ground Based Concept of Operations

3. Critical Project Elements

KESSLER is comprised of four main Critical Project Elements (CPEs) which are: paramount to achieving project objectives, will most likely require a lot of time and effort, and/or also have a level of technical uncertainty. The Robotic Arm (RA) and Visual Processing (VP) aspects of the project are key constraints required of the system. The 4 CPEs KESSLER identified are all technical aspects since resources and monetary concerns are not of high risk based on the current concept design.

CPE 1: Mechanical (RA): The mechanical design and dynamic actuation capability is one that is critical to not only demonstrating successful capture capabilities but also is critical in bounding the scale (size) of the project. Monetary considerations have been made to identify a feasible solution for the robotic arm. The arm operates with 6 DOF in order to maximize its range of motion so that it may grab features on the object at arbitrary angles. The arm will also be able to re-position to get more complete data and that will aid the visual processing segment recognize features that may not be visible from the initial position.

CPE 2: Control Software (RA): The control software will have two major tasks as the project progresses. The first is related to the robotic arm and involves implementing additional capability that did not get fully developed by the heritage team. The second is to add on to the existing software an understanding of the position of obstacles and path determination that will allow the arm to avoid it.

CPE 2.1: Arm Control: The control algorithm created by the heritage team met their primary objectives but had areas that could be improved on to increase the range of motion capabilities. In order for KESSLER to meet the project capabilities this area needs to be addressed. Specifically with the arm's ability to capture an object using the wrist bend of the claw. Although the mechanical capability is available for this additional range of motion the current software cannot directly resolve the encoder inputs and command the actuators to the desired position without reaching an indeterminate state in the control algorithm.

CPE 2.2: Path Determination: Path determination was demonstrated on the heritage project to capture within the plane of of the arm's mounting base plate. In order to meet the customer objectives this year, KESSLER must be capable of implementing path determination that involves grappling at various angles with respect to the arm's baseplate.

CPE 3: Electrical (RA/VP): This element is comprised of key support electrical equipment, electrical design required for the visual processing subsystem, and electrical design required for the robotic arm. Key electrical harness, connections, critical electrically driven components that take command input from software and actuate the desired motion (i.e. motor drivers, motors, etc.) as well as take data from sensors for the software to then interpret (motor encoders, and cameras). A significant challenge of this system is the flex cable design which contains all electrical wiring throughout the arm and must support the ROM of the arm. Additionally, if KESSLER's algorithms require high-speed image processing capabilities, in-depth knowledge of embedded system design which requires a substantial amount of electronics background will be necessary.

CPE 4: Feature Recognition (VP): This element is required to satisfy the project objectives for feature recognition. This area is one that is not included in the ASEN coursework and requires a deep level of knowledge in computer science. Although a significant percentage of the KESSLER team has a computer science background, this element will require substantial background research of how image processes and machine learning fundamentally works before developing on the image processing algorithm for KESSLER.

III. Design Requirements

A. Flow-Down Process Approach

In order to develop the requirements flow down from the functional to design requirements the functional requirements were revisited since the Preliminary Design Document (PDD) phase. The primary feedback obtained from the first attempt to functional requirement definition is the fact that the requirements didn't have a sufficient amount of metrics that provided confidence in determining a validation method for them. The KESSLER team researched additional information to help bound the project requirements further (in collaboration with the customer). A significant amount of the design requirements were then obtained by deriving additional metrics from the functional requirements and accounting for capabilities that from preliminary research, are achievable (the project team also attempted to flow down a third requirements level as more information was gathered). These requirements will be assessed for feasibility as the project progresses towards the Preliminary Design phase.

B. Requirements Flow-Down

F1. Robotic arm shall utilize visual processing system to locate target object in its line of sight to within a 30.5 inch (77.47 cm) linear distance from mounting base to the closest edge of said object. (*Length corresponds to maximum Extension of Arm, and visual processing are both customer constraints*)

F1.1. The visual system shall identify the location (x,y,z) and orientation (Euler angles) of an object in 3D space. (Derived)

Motivation: To determine the location and orientation of the target object with respect to the base plate.

Validation: Measurement of baseplate to location (use a distance measuring device (protractor, ruler, VICON))

F1.1.a. The system shall determine a body coordinate frame an origin of the target object.

F1.1.a. The system shall identify feature edges to within XX cm.

F1.1.a.i The system shall have a Field Of View (FOV) such that there's a horizontal bound range of 35.2 - 42.7in and a vertical of 28.4 - 35.2in (Derived) at 30.5in.

F1.1.b. The system shall use an RGB sensor for visual processing. (*Customer Constraint*)

F1.2 The visual system shall be capable of communicating with the control system. (Interface).

Motivation: The control system needs the location and orientation of target object in order to translate into actuator commands.

Validation: Transmit and receive data packets will be identical.

F2. The robotic arm shall capture at least one pre-selected grappling feature on target object. (*Customer Constraint*)

F2.1 The robotic arm shall be capable of receiving commands from the control system. (Derived).

Motivation: The control system must be able to take inputs and transmit to physical movements of the mechanical arm.

Validation: Develop test commands and verify (through inspection) that the actuators executed command.

F2.1.a The robotic arm shall be able to initiate operations based off of a command from the CPU. (Derived).

F2.1.b The robotic arm shall terminate operation upon command from the CPU. (Derived).

F2.2 The grappling feature shall be representative of common features found on the Iridium Constellation Satellite form factor. (Derived).

Motivation: Satellites currently prevalent in LEO and are well documented with the well known grappable features (approved by customer).

Validation: In-House manufactured grappable features coincide with Iridium Satellite feature form-factor (inspection).

F2.2.a. The Iridium Constellation Satellite shall be scaled by 0.60.

F2.2.a.i. The feature shall be an existing object on the satellite with a grappable thickness no greater than 8 inches (20.32 cm). (Derived).

F3. Positioning algorithm shall provide a path for target location to capture target object.

F3.1. The end-effector orientation and locations shall be determined in 3D space to within $\pm$13mm and $\pm$5deg.

Motivation: This is derived based on the maxim arm extension and worst case angle error in motor encoders with a FOS of 2. The worst case angle error is a sum of the angle error for each motor times the number of motors.

Validation: Measurement of baseplate to location (use a distance measuring device (protractor, ruler, VICON)).

F4. The robotic arm shall autonomously grasp feature on target object.

F4.1. Robotic arm shall move in path outlined by positioning algorithm.

Motivation: The arm will be able to move on a predetermined path.

Validation: Verify with positioning sensor to compare actual path to predetermined path.

F4.2 The end effector shall be able to capture objects of (F2.2.a) size.

Motivation: Customer constraint.

Validation: By visual inspection.

F4.2.a. End effector shall have a fully deployed range of no more than 9 inches.

F4.2.b. End effector shall secure object without compromising structure of grappled object.

F5. F5: The KESSLER system shall have a total process operations time defined by the Sun-Soak phase of an average LEO orbit (total mission phase 45 minutes $\pm$10 minutes).

F5.1 The KESSLER system shall be capable of successfully capturing object at 2 of 3 attempts during operations.

Motivation: Total operational time needs to be defined. Derived from customer constraints.

Validation: Measure time duration from image capture to feature capture and ensure it is below 15 minutes.

F5.1.a Image identification, grappling maneuver, and capture will take no more than 15 minutes to be executed.

IV. Key Design Options Considered

A. Design Options Overview

Prior to approaching trade studies, the KESSLER project is divided into 4 primary subsystems: Visual Processing, Control Software, Mechanical Robotic Arm, and Ground & Test Support. Although each one of these subsystems are depend on one another, they each have primary technical areas that needed their own trade studies to then compile the selected options to then define the baseline design.

Visual Processing is comprised of three primary technical areas which are: Primary imaging hardware (which addressed they key customer constraint of using an RGB sensor for feature detection), software (which is the means of which the image data is translated into a target location), and secondary imaging hardware (which is used to address fine adjustments once the robotic arm has maneuvered near the feature to grapple - this was added as an area since it will provide an additional level of risk reduction for grappling location determination).

Control Software is comprised of three primary technical areas which are: control software (which is the means to interpret the data from the motor encoders and commanding the motors to actuate), thermal monitoring (which is a method to address the lesson learned of being cautious of overheating the actuator(s)), and software integration platform (which is the area that predicts the path the robotic arm should take based on the data sent from visual processing and the encoder data).

Mechanical Robotic Arm is comprised of 3 primary technical areas which then 'feed' into the configuration selection trade. The primary 3 are: robotic claw (which impacts grappling range ROM), the girder length (which impacts the arm's maximum length), and actuators (which addresses significantly increasing the ROM of the arm to grapple at various angles with respect to the base plate, since another DOF is to be added to the arm, an actuator for that joint needs to be selected). The final area is the trade between four design configurations which were developed in combination with the aforementioned 3 technical areas.

Ground and Test Support is comprised of two major areas which help in scoping the key spacecraft size (and model) that should be used. This then drives into the trade in which at least 3 types of grappling features are selected from a group 5 to further define the project functionality.

A visual overview of these subsystems and their trades can be seen in Figure 6. The various interfaces between these different subsystem (and other subsystems that were not individually prominent for the concept design study to have their own section) can be seen in Figure 7. The arrows connecting each subsystem module provides an indication of how the various subsystems needed to relay information as their individual trades developed to reach a final consensus.

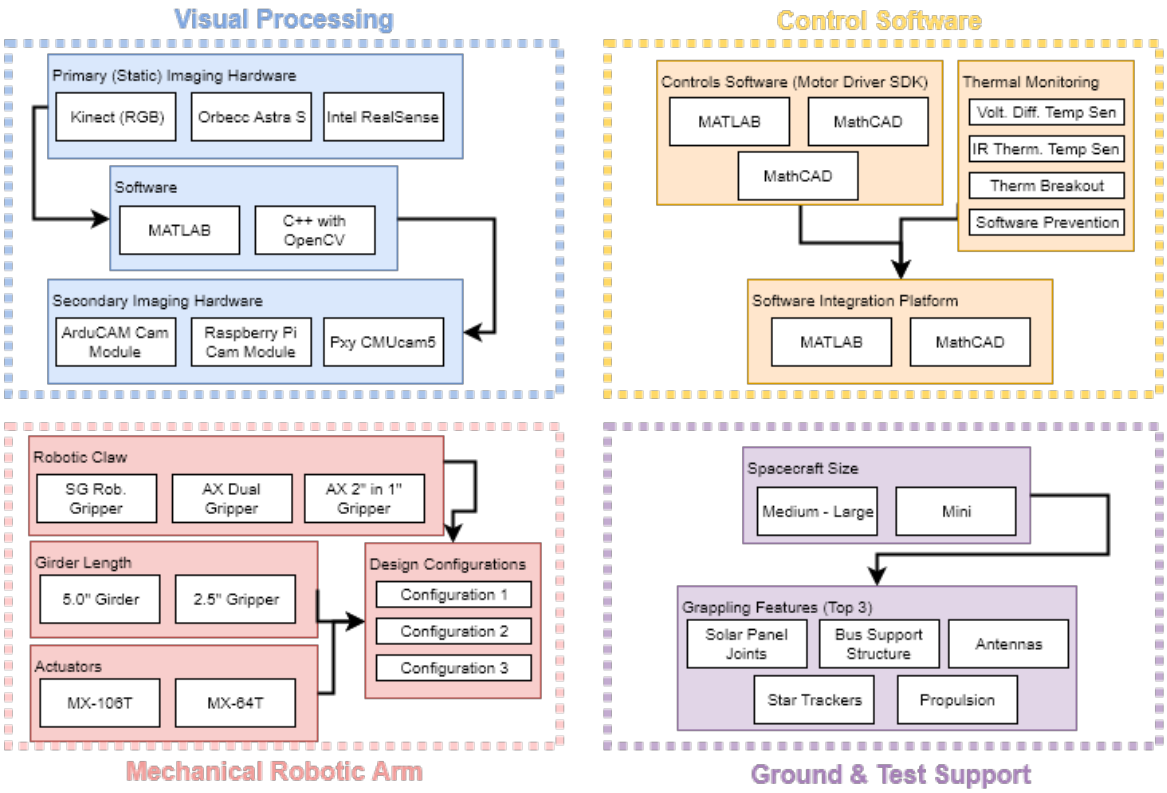

Figure 6: KESSLER Subsystems Trade Flow Diagrams

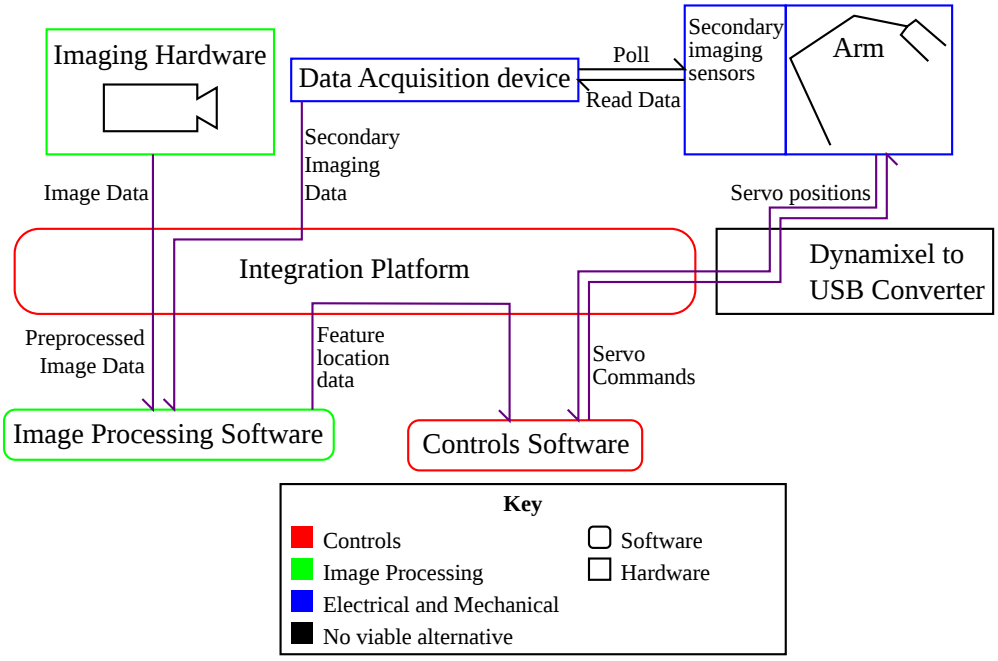

Figure 7: KESSLER Key Area Trade Interface Flow Diagram

B. Visual Processing: Imaging Hardware

The primary function of the imaging hardware is to provide information on where the arm and satellite are located. This hardware will consist of an RGB camera, as per customer requirement, that with software will be able to locate various features on the satellite that are appropriate for grappling. This hardware is also to be used to combat the issue of the robotic arm drooping two inches lower than where it thinks it is located due to its weight and gravity.

The three types of hardware that are considered for this function are the Microsoft Kinect sensor, Orbbec Astra S sensor, and Intel RealSense SR300 sensor. These sensors are all very similar and used for motion tracking. The Kinect sensor was used by the CASCADE team and has been passed down to the KESSLER team. While each sensor is similar, the Kinect has the most documentation online and users that use it for tracking purposes. Other than this, the Orbbec Astra S sensor has the same RGB sensors as the Kinect and similar interfacing. The last sensor, the Intel RealSense SR300, has the same RGB sensor but a much more stringent interface which has gone out of service by Intel. Below is a detailed description of each sensor and a trade study to choose the best sensor for KESSLER.

1. Microsoft Kinect

The Microsoft Kinect is a motion sensor add-on device for the Xbox 360/One gaming console. The Kinect enables users to use a natural user interface (NUI) as a method to interact with the gaming console without the need of a controller. The Kinect was first released in 2011, and has developed a community of developers that can utilize the Kinect hardware to create custom solutions for various applications [3].
The hardware on the Kinect is comprised of three different areas seen below [4]:

- Color VGA video camera - facial recognition and other detection features by detecting 3 color components: red, green and blue (RGB). Pixel resolution: 640 x 480 at 30 frames per second (FPS). Field of view (FOV): 70x60 degrees [6]
- Depth sensor - IR projector and monochrome CMOS (complementary metal oxide semiconductor) sensor work together to 'see' the room in 3D regardless of lighting conditions. Pixel resolution: 640 x 480 at 30FPS
- Multi-array microphone - Four microphone array that can isolate users' voices from noise in room. Console commands can be made by users' that are a few feet away from the console.

For the purposes of KESSLER, the first two hardware areas are readily applicable to meet the functional, design, and customer requirements. The Depth Sensor on the Kinect was utilized by the heritage project CASCADE which not only provides a first step in obtaining partial functionality for KESSLER but also a level of confidence in the hardware (and accompanying software) to be approachable.
The Kinect hardware is easily accessible for developing custom applications via the Software Development Kit (SDK) Microsoft released. SDK Includes Windows 7 compatible PC drivers for Kinect device. Provides capabilities to develop applications with C++,C# or Visual Basic by using Microsoft Visual Studio 2010, this includes the following features/capabilities [4]:

- Raw sensor streams: access to low-level streams from the depth sensor, color camera sensor, and four-element microphone array
- Skeletal tracking: capability to track skeleton image of one or two people moving within Kinect's field of view for gesture-driven applications
- Advanced audio capabilities: (data not included since this will not be relevant to our project)
- Sample code and Documentation

The Image Acquisition Toolbox in MATLAB also has a hardware add on package for the Kinect that makes it easy to interface the Kinect through MATLAB [5].

Figure 8: Kinect sensor

Table 2: Pro/Con list for Microsoft Kinect sensor package.

Pros	Cons
Low Cost Solution with Depth and RGB sensors	Requires Windows Operating System
SDK available and well documented	
On-campus resources have experience with developing on Kinect (Dan Godrick)	
Heritage hardware provided by customer	
Can be interfaced with MATLAB	

2. Orbbec Astra S

The Orbbec Astra S is a low cost RGB sensor option that specializes in 3D scanning. This sensor specifically has a depth camera and an RGB camera. It is optimized for short range 3D scanning with the best images from a distance of 0.4-2 meters. The sensor is also fully compatible with OpenNI, an open source software that is used to track user movement. The sensor also comes with a gestural interaction software that allows for development in C++, Java, and Processing. Additionally, OpenNI can also be easily interfaced with MATLAB for image analysis [7].

The pixel resolution for the RGB camera is 640 x 480 at 30 FPS for a normal image and 1280 x 960 at 7 FPS. Additionally, the RGB camera also has a FOV of 60 x 49.5 degrees [8]. Data transfer with the sensor is simple and is transferred through a USB 2.0. However, the depth sensing for this sensor is only compatible with Windows but the RGB sensor itself is compatible with Windows, Linux, Android, and OS X [7].

Figure 9: Orbecc Astra S sensor

Table 3: Pro/Con list for Orbbec Astra S sensor package.

Pros	Cons
Used for 3D scanning	Depth sensor is Windows only
Can be interfaced with C++ or MATLAB	Primarily used for tracking human movement
Low cost	
Simple data transfer	

3. Intel RealSense SR300

The Intel RealSense SR300 is an integrated 2D and 3D sensor. The sensor features an RGB camera, an infrared camera, and an infrared laser projector. This camera is optimized for streaming and video conferencing and can only run on Windows using the Intel RealSense SDK software. The SDK software also supports dynamic background segmentation, 3D scanning, and facial recognition and gesture recognition [9]. This sensor also has a FOV of 73 x 59 degrees [10]. Lastly, data transfer is easy for this sensor as it uses a USB 3.0 to transfer data from the sensor to a Windows OS [9].

A new version of this sensor is coming out in October 2017 and will replace the older SR300 model. Since this version of the RealSense sensor is being discontinued the software that it uses is also being discontinued and will no longer be supported [11].

Figure 10: Intel RealSense SR300 sensor

Table 4: Pro/Con list for Intel RealSense SR300 sensor package.

Pros	Cons
Integrated 2D and 3D sensor	Sensor is being phased out of Intel
Low cost	Must use Intel RealSense SDK software
	Only works on Windows

C. Visual Processing: Secondary Visual Imaging Hardware

Due to the primary visual processing hardware being mounted external to the arm, a secondary visual sensor is necessary to provide visual data from the point of view of the claw. This is two-fold; first to verify the location accuracy of the initial image processing and execution and second, to have a verification image for the grappling feature. Primarily, research was done to have any kind of proximity sensor, this was narrowed two having a secondary camera upon meeting with our customer. The image would be able to provide not only proximity but finer tuning of grappling feature identification. It would also be able to provide visual confirmation of successful grappling.

1. ArduCAM Mini Camera Module shield with 2 Mega Pixel OV2640

The first hardware unit compared is a high definition SPI camera. It uses a 2 MP CMOS image sensor equipped with small package dimensions, easue to use hardware and open source code library. It is designed to work well with an Arduino Microcontroller. In addition, it supports a variety of operation modes to suit the need of the user, in our case supporting single capture and low power mode. The image sensor allows for up to 2 megapixel capture, both I2C and SPI interface with the controller, multiple lense options and an active array size of 1600 x 1200 pixels.

Table 5: Pro/Con list for ArduCAM Mini Camera Module.

Pros	Cons
2 MP image sensor	No internal image processing
easy to use hardware interface	
open source code library	
can be used in any platforms like Arduino, Raspberry Pi, etc	

2. Raspberry Pi Camera Module V2

The next hardware unit considered runs exclusively on Raspberry Pi. It operates at up to 8 MP capable of 1080p video and images. It is a tiny 25mm x 25mm x 9mm and just over 3g which will not remotely hinder the operation of our arm. For our purposes we will employ the 3280 x 2464 pixel static image functionality of the device. The image sensor is a Sony IMX219 that connects using CSI on a 15cm ribbon cable and is supported by Raspbian OS. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.

Table 6: Pro/Con list for Raspberry Pi Cam Module V2.

Pros	Cons
8 MP Sony IMX219 Camera	No internal image processing
Ultra small and lightweight design	Only Raspberry Pi OS
high data transfer rate	Fixed Focus Image Sensor

3. *Pixy CMUcam5*

This absolutely remarkable piece of equipment can be taught to find objects of varying shapes, sizes and colors. The Pixy is incredibly fast and easy to use with its dedicated processor, NXP LPC4330. This processor permits the device to handle large loads of data and process it quickly, outputting results 50 times per second. Pixy operates on a variety of connectors; UART serial, SPI, I2C, USB or D/A output making it very versitile on microcontroller interface options. The sensor itself is the Omnivision OV9715 quarter inch lense capable of 1280 x 800 pixel resolution. It has onboard RAM of 264 Kbytes and 1Mbyte flash. It comes in bulkier than the previous options but not overbearingly at 2.1" x 2.0" x 1.4" weighing 27 grams. Finally it comes with extensive documentation, quickstart guides and troubleshooting material.

Table 7: Pro/Con list for Pixy CMUcam5

Pros	**Cons**
On board image processor	substantially larger than other options
Lens field-of-view 75° horizontal and 47° vertical	significantly more costly
can be taught to recognize various shapes, sizes and colors	
Quick 50 times a second output	

D. Visual Processing: Software

The visual hardware that is implemented on the robotic arm will need an accompanying software to analyze the image and tell the arm where the optimal grappling feature is located. This software should integrate easily with the other software used by the KESSLER team and be user friendly for future teams.

The two software option that are considered for the visual processing are MATLAB and C++ with OpenCV. MATLAB features an Image Processing toolbox that comes with many built in functions specifically made for image processing and visual recognition. MATLAB is also being used as the primary software for KESSLER and is relatively user friendly and known by the entire team. C++ with OpenCV, while a powerful compiler, does not have many built in functions for visual processing. It is also much less user friendly and not the primary software platform for the robotic arm.

1. *MATLAB*

MATLAB is short for Matrix Laboratory and is used for scientific calculations and I/O applications. It has many built-in functions, as well as many available toolboxes for various disciplines. A couple of toolboxes relevant to KESSLER's visual imaging which include the Image Processing toolbox and the Computer Vision System toolbox. With or without these toolboxes, it is possible to: import, display, and annotate images and videos; detect, extract, and match object features; and detect objects in images and videos [12]. Using the toolboxes provides many functions to make such processes easier, but the toolboxes are not necessary.

Objects are detectable in an image through pattern recognition algorithms and deep learning, which can be made possible using the Statistics and Machine Learning Toolbox. Feature-based object detection can be used through the use of feature points. A reference image is used to obtain these points, and this "method of object detection can detect reference objects despite scale and orientation changes" [13]. A pictorial example of feature matching in MATLAB can be seen below in figure 11.

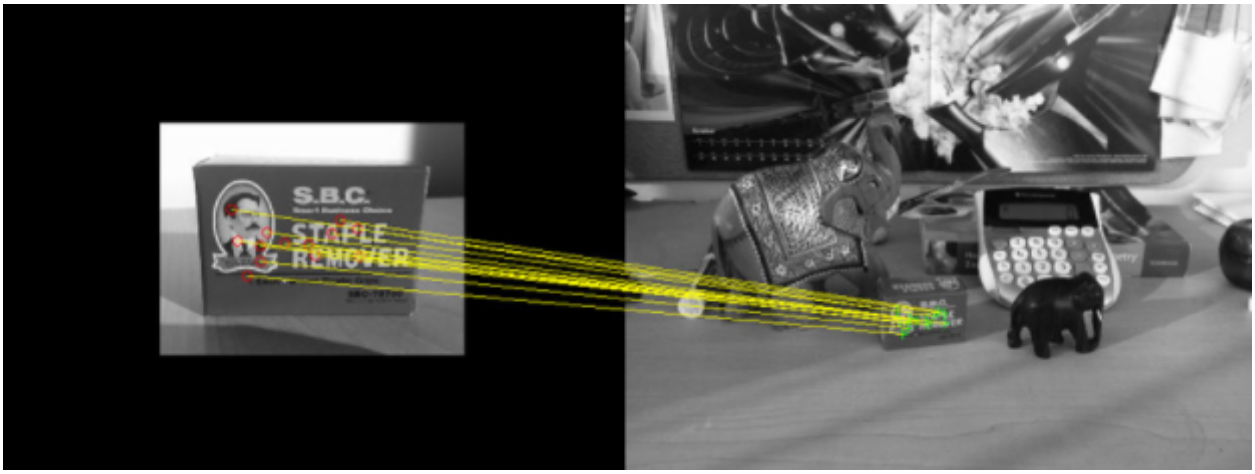

Figure 11: Example of MATLAB's visual processing power.

Table 8: Pro/Con list for MATLAB as a software option [14]

Pros	**Cons**
Powerful matrix library	Slow runtime
Toolboxes (image processing, computer vision, statistical and machine learning, optimization, etc.)	Software and toolboxes are expensive
Visualization and debugging tools	
Works with OpenCV	
Great documentation	

2. C++ with OpenCV

C++ is an object-oriented programming language that was developed as an extension of the C language. OpenCV (Open Source Computer Vision Library) has interfaces for C++, C, Python, and Java. It was designed for "computational efficiency and with a strong focus on real-time applications" [15]. Using OpenCV provides a library with many free algorithms. OpenCV is optimized for use with C++.

Additionally, C++ is a compiled language, making it very fast and efficient if optimized. It is one of the most frequently used languages and an open language, so a wide range of compilers can be used. Likewise, it can be used among platforms with few or no changes to the code [16].

Table 9: Pro/Con list for C++ as a software option [14]

Pros	**Cons**
Largely free	Difficult for beginners
Huge optimized library	Weak documentation
Platforms and devices (used in many embedded vision applications)	Small set of machine learning algorithms
Large community of developers (for help)	Difficult visualization and debugging
Fast runtime	

E. Control Subsystem: Software Integration Platform

The software integration platform will allow the team to develop visual processing modules and control modules by commanding/obtaining data from the vision sensors and servos. It will also be used to develop the algorithms to obtain the position and orientation of the end-effector using the collected data. It is important that the language of the chosen platform can be used to effectively command/obtain data from vision sensors and servos. Centralizing the programming languages used will help to reduce the learning curve and optimize efficiency of the software team. Two main options will be considered: MATLAB and C/C++. Both of these are common in the aerospace industry and

each language provides unique capabilities. MATLAB is useful for algorithm prototyping. MATLAB also includes computer vision tools and has the capabilities to bridge/interface with many computer vision tools that are written in C/C++ like OpenCV, Point Cloud Library, and CUDA. On the other hand, code written with C/C++ is optimized for speed and memory efficiency. Code is often translated to C/C++ during production to maximize performance.

1. MATLAB

MATLAB is a programming language that handles matrix math, machine learning, function plotting, visual processing, and includes many other capabilities. It is commonly used in the aerospace industry as well as in university aerospace programs. It is user-friendly and easy to debug. Many pieces of hardware, as well as other software platforms, can interface easily with MATLAB.

MATLAB also has easy-to-use serial communications packages [24] that are able to interface with any data acquisition device that supports serial protocols. One such device is the Arduino microcontroller. These popular microcontrollers are cheap and simple to program using a modified version of C++. Importantly for this trade, MATLAB has libraries that are optimized for communication with an Arduino. [25]

Table 10: MATLAB Pros and Cons

Pros	**Cons**
Legacy code	Slower run time
Prior team knowledge	Uses more memory
Available in industry	
Good visual processing capabilities	
Good online documentation	
Great matrix library	
Easy to debug	
Can interface with OpenCV	

2. C/C++

C++ is an object-oriented programming language that is used to write general applications. It is prevalent across many industries, including aerospace. C/C++ has been developed to interface well with numerous other languages, and there is a long history to the language. It is well documented and has a long online following. Because of its speed and capabilities, it will also be considered in a trade study.

Table 11: C/C++ Pros and Cons

Pros	**Cons**
Prior team knowledge	Harder to debug
Optimized libraries, fast to run	No inherited code is in C/C++
Some visual processing libraries	Not as good for linear algebra
Good online documentation	More difficult to learn
Available in industry	

F. Control Subsystem: Controls Software

This subsystem area covers how the arm translates commands that tell it to go to a certain location into the individual motor movements that lead to this movement. This can be broken down into two major areas. First, receiving data from and sending commands to the Dynamixel servo motors that move the arm is a core requirement. The second is the calculation of commands by solving systems of equations.

The data from the servos on the robotic arm will be used to calculate the position and orientation of the end-effector. Since the manufacturer of the servos have already written a software development kit (SDK) that includes libraries and functions used to obtain data from the servos, we will be looking at which software development kits are appropriate for our purposes. There are three SDKs that we will consider, one in LabVIEW, one in MATLAB, and one in C++. The main difference between them is that heritage work was done to create velocity command functions in MATLAB for smooth movements of the arm.

In order to calculate these commands, the code must solve complex systems of equations that current algorithms implement using matrix mathematics. These systems require the control software system to have effective algorithms for these operations. This is an additional metric by which these trades can be assessed.

1. Dynamixel SDK- LabVIEW Version

LabVIEW is an integrated platform for designing software using graphical language. The flow of the software in design is determined by the structure of a graphical block diagram in LabVIEW that the user can draw using a palette of functions. LabView is optimized for hardware interfaces, including driving motors and reading sensors. Most of these functions are optimized through associated hardware from National Instruments. The Dynamixel SDK has a full Application Programming Interface (API) open to LabView [21]. The LabView implementations of the API functions read from a shared library of precompiled C++ code to execute their functions.

In order to calculate the motor adjustments necessary to move the arm, complex matrix operations are required. LabView has linear algebra functions that may be sufficient to meet this requirement. However, due to the format of LabVIEW's graphical code, these calculations may be difficult to debug, with data lines crossing and not having clear sources.

Table 12: Dynamixel SDK- LabVIEW Pros and Cons

Pros	**Cons**
Prior team knowledge	Team not familiar with
Easy to learn because graphical programming	Use C as binding

2. Dynamixel SDK- MATLAB Version

The Dynamixel SDK has a full API open to MATLAB. MATLAB interfaces well with C, C++, and things derived from those languages. For C and C++, it can run these files pretty much directly by compiling them into "mex-files". In the case of the SDK, the MATLAB implementations read from shared precompiled modules of the C++ code that is the backbone of the library.

MATLAB's primary strength is in its ability to efficiently and intuitively manipulate matrices. This is a large benefit in the consideration of using it as the language for creating the control algorithms.

Table 13: Dynamixel SDK- MATLAB Pros and Cons

Pros	**Cons**
Prior team knowledge	Latency
Heritage work	Use C as binding
No learning curve	

3. Dynamixel SDK- C++ Version

C++ is an object-oriented language based on C. It is commonly used for general programming. It has a long history and widespread support. Popular compilers such as gcc (the GNU compiler collection) and Visual C++ for Windows support it. The Dynamixel Software Development Kit (SDK) was originally written in C++, though it has been fully ported to C as well. Other languages (MATLAB, Labview, Python, and Java) are also supported, directly importing the C/C++ code libraries through their own capabilities. The API for C++ naturally supports all the functionality of the full library. However, some of the functions are platform-specific (linux or Windows) and therefore portability is a little lacking. From a cursory view of the code, most of these portability problems can be resolved in a fairly simple manner, as they simply involve interchanging function names.

There are open source linear algebra libraries written for C++ (like Armadillo [26]) which contain algorithms to implement many common matrix operations. This may also satisfy computational requirements.

Table 14: Dynamixel SDK- C++ Pros and Cons

Pros	**Cons**
SDK backbone is written in C/C++	Harder to debug
	No inherited code is in C/C++
	More difficult to learn

G. Control Subsystem: End Effector Thermal Monitoring

One of the known problems on the CASCADE project was the overheating and malfunction of the end effector servo. This malfunction occured whenever the servo was continually actuated for over 4 minutes (a requirement of the CASCADE project). The KESSLER team will solve this problem in order to avoid damage to the servo due to overheating during operation. There are 2 main options being investigated: thermal sensors to trip a fault in the control code to shut down servo operation, and adding a timer into the control code to avoid the 4 minute actuation limit. Three different sensors will be compared along with the software module option.

1. Voltage Differential Temp Sensor

The first option is a voltage differential temperature sensor. The sensor can be as small as 5.5 mm on the widest dimension and can be as accurate as $\pm$ 2 degrees C over the -40 to 125 degrees C range. Upon contact, the device outputs a voltage that is linearly proportional to the Celsius temperature scale and is powered with 2.7-5.5VDC. These sensors are very low cost and can be as low as 1.5 USD each. This sensor can be attached to the end effector's servo to function as a failsafe in the event that the servo overheats.

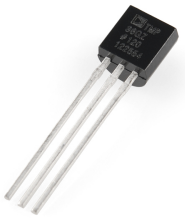

Figure 12: Example of a voltage differential temperature sensor

Table 15: Voltage Differential Temp Sensor Pros and Cons

Pros	Cons
Small and light weight	Need data acquisition device
Small cost	Need wiring on the arm
	Need to determine mounting location

2. Infrared Thermopile Temp sensor

The Infrared Thermopile Temperature sensor can measure the temperature of an object without making contact with the object by absorbing the infrared engergy emitted by the object. It runs on 3.3-5VDC, has a field of view of 90 degrees, and temperature sensor resolution of $\pm$0.03125 degrees C over the -40 to 125 degrees C range. This sensor costs around 15 USD each and works with most microcontrollers via the I2C bus.

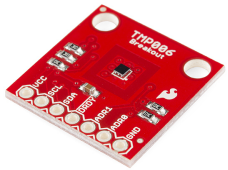

Figure 13: Example of an infrared thermopile temp sensor

Table 16: Infrared Temp Sensor Pros and Cons

Pros	Cons
Doesn't need to be touching servo	Need data acquisition device
	Need wiring on the arm
	Need to determine mounting location

3. Thermocouple Breakout

The Thermocouple Breakout is a simple 14-bit resolution breakout board with an accuracy of ±2 degrees C from -200 to +700 degrees C. It requires a SEN-0025 Ktype thermocouple (about 7in long) with the cold junction kept cool while the hot junction is placed in contact with the object measured. It uses a power source of 3 to 3.6 VDC. This device costs around 15 USD each.

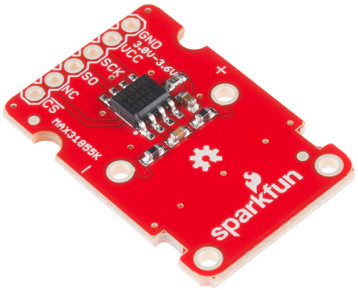

Figure 14: Example of a thermocouple breakout board

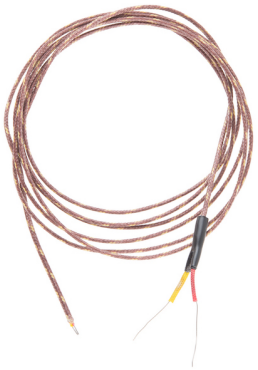

Figure 15: Example of a thermocouple

Table 17: Thermocouple Breakout Pros and Cons

Pros	Cons
Great range!	Need data acquisition device
	Need wiring
	Need to determine mounting
	Need thermocouple and thermocouple connector

4. Software Prevention

A software fix can be added into the current control algorithm to prevent it from continuously actuating the servo for more than 4 minutes. Once actuation of the servo begins, a timer will also begin. The new algorithm will pause the servo for a period of time in the event the actuation time has gone over the 4 minute threshold.

Table 18: Software Prevention Pros and Cons

Pros	Cons
Easily integrated to software	Need to determine pause time and threshold
No hardware required	
No physical size or weight	

H. Mechanical Robotic Arm: Robotic Claw

The primary function of the robotic arm is to physically implement the positioning algorithm and move to a predetermined location in order to grapple a feature on an object. The robotic arm is inclusive of the grappling claw,

the actuators which drive the movement of the arm, and the girder extensions. These parameters effect the range of motion, extension, and rotational power of the integrated system.This particular study focuses on the robotic claw which enables the robotic arm to grapple an identified feature on the target satellite. There are three commercial off the shelf claws which are (supplied) by CrustCrawler Robotics, the SG Gripper, AX Dual Robotic Gripper, and the AX 2” in 1” Robotic Gripper. Both the SG and AX 2" in 1" grippers run off of a single actuator and have a smaller grappling range while the AX Dual Gripper has the largest range and operates on dual actuators.

1. SG Robotic Gripper

The SG Robotic Gripper is one of three gripper mechanisms sold off the shelf by CrustCrawler Robotics. It has the ability to integrate to any standard sized servo, giving the end user an augmentation of both gripping strengths and movement capabilities. The SG Gripper also has the capability of adapting ultra sonic sensors to enable the proximity identification of nearby objects. The dual rotation and easy adjustment of the individual plates makes this gripper a versatile option for grappling.

Table 19: SG Robotic Gripper Pros and Cons

Pros	**Cons**
Can accomodate any standard sized servo	Cannot extend linearly
Increased surface area contact of feature	
Capable of grappling object with additional force	
Large range of motion of gripper plates	

Figure 16: CrustCrawler SG Robotic Gripper

2. AX Dual Robotic Gripper

The AX Dual Gripper is a versatile robotic gripper which serves as an extension of the CrustCrawler Pro Series mechanism. The AX Dual Gripper has a maximum open claw extension of 22.86 cm and has the capability of integrating to both AX-12A and AX-18A servos. The AX Dual Gripper serves as heritage hardware used by the 2016-2017 CASCADE senior projects team. It successfully completed a slew of tests to grapple a 3U CubeSatellite for the nominal time of 5 minutes. Extensive documentation on both the system’s characteristic and previous testing with the CrustCrawler integrated unit is available.

Table 20: AX Dual Robotic Gripper Pros and Cons

Pros	**Cons**
Hardware available in-house	
Heritage hardware that is well documented	
Most versatile gripper opening (clamping range)	Reduced surface area contact
Accommodates two types of servo motors	Servo may not fit project need

Figure 17: CrustCrawler AX Dual Motor Robotic Gripper

3. AX 2" in 1" Robotic Gripper

The AX 2" in 1" Robotic Gripper is the last of three robotic grippers sold off the shelf by Crust Crawler Robotics. It has the largest footprint of the three available grippers which enables the system to grapple a wide array of objects from small to large in a form factor similar to that of the dual robotic gripper. The AX 2" in 1" has a set of small and large parallel plates which enable the

Table 21: AX 2" in 1" Robotic Gripper Pros and Cons

Pros	Cons
Versatile clamping mechanism	
Capable of gripping large and small objects	Large and adds weight to structure
Opening of one clamp	One point failure

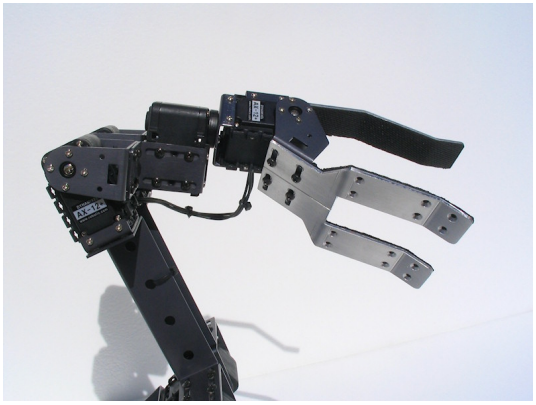

Figure 18: CrustCrawler AX 2" in 2" Robotic Gripper

I. Mechanical Robotic Arm: Sixth Degree of Freedom

The purpose of this trade study is to determine the optimal location and power requirements for the addition of a DOF to our current 5 DOF CrustCrawler Robotic arm. The study will take into account the placement of the new actuator, the type of actuator to add based on power and weight requirements, and type of "Girder" extender to add based on weight and length requirements. Four separate scenarios will be evaluated for efficiency, feasibility, cost, and performance in order to select the most practical location, actuator, and "Girder" for the new DOF.

When adding a new degree of freedom is it vital to take into account which section of the arm to add it to. The addition of the actuator should be seamless and must fulfil the given power and length requirements while maintaining optimal arm performance. Since the current arm is pretty well designed towards the claw end it would make more sense to add the actuator towards the base. Adding the actuator closer to the base of the arm will maintain arm performance

and range towards the claw end while seamlessly adding extra flexibility to the overall mechanism. Adding the actuator to the end of the arm (by the claw section) would sacrifice performance and accuracy and would require a complete redesign of that section's actuators and Girders.

1. Mechanical Robotic Arm: Range of Motion

It is important to note that the KESSLER team opted for keeping the AX Series CrustCrawler Robotic Arm due to the modular nature of the assembly which allows the user to easily integrate different actuators and extensions on the structure. Keeping the heritage hardware also reduces the cost of the assembly and allows for a streamlined process to characterize the system due to the availability of last year's documentation of both software and hardware developments.

The purpose of this trade study is to determine the optimal location and power requirements for the addition of a DOF to our current 5 DOF CrustCrawler Robotic arm. The study will take into account the placement of the new actuator, the type of actuator to add based on power and weight requirements, and type of "Girder" extender to add based on weight and length requirements. Four separate scenarios will be evaluated for efficiency, feasibility, cost, and performance in order to select the most practical location, actuator, and "Girder" for the new DOF.

When adding a new degree of freedom is it vital to take into account which section of the arm to add it to. The addition of the actuator should be seamless and must fulfil the given power and length requirements while maintaining optimal arm performance. Since the current arm is pretty well designed towards the claw end it would make more sense to add the actuator towards the base. Adding the actuator closer to the base of the arm will maintain arm performance and range towards the claw end while seamlessly adding extra flexibility to the overall mechanism. Adding the actuator to the end of the arm (by the claw section) would sacrifice performance and accuracy and would require a complete redesign of that section's actuators and Girders.

The selection of actuator to add is extremely important. The cost and power must be taken into account so as to maximize efficiency and performance. The power requirements of the actuator is determined by the placement of it; the actuator must be able to hold and move the sections that are ahead of its placement locations. An underpowered actuator could cause critical mission issues if it is unable to handle the weight of the section it is meant to hold/operate. Additionally, modifications to existing actuators must be made depending on the placement of the new one. If the new actuator is placed in front of an existing one, a study must be done on the existing actuator to confirm that it is capable of holding the added actuator section given its current power rating.

Finally, the selection of Girder length is important as it dictates the range and mobility of the new system. The location, weight, length and mobility requirements, and surrounding Girders must all be taken into account in order to achieve optimal performance. The length of the Girder also depends on the placement of the actuator and its surroundings as it needs to be seamlessly integrated given range requirements. The arm needs to be flexible enough to easily reach around (not behind) the object it is grappling. It is important to choose only the needed length as added length could create weight issues and in turn cause unnecessary changes to the system.

J. Mechanical Robotic Arm: Girder Length

The heritage CASCADE arm has a max extension length of 22 inches (measured from the base of the Dual Dynamixel Actuators to the base of the robotic claw). This allows the arm to capture any object within a semi-sphere range with a 22 inches radius. However, due to its clustered actuator design, the arm currently has difficulty grappling objects using a non-planar approach (i.e. from the side, from the back). Therefore, the current design limits the grappling capabilities of the arm at non linear orientations with respect to the mounting platform. In order to maximize the arm's range and grappling capability, an extra degree of freedom will be needed. the added range will give the arm a wider Range of Motion (ROM) by expanding its Sphere of Range from a 22 inch radius hemisphere to that of a 28 - 30 inch radius.This increased ROM will be achieved by the addition of a girder of either 2.5" or 5.0".

1. 2.5" Girder

The first of two options for extending the arm involves the addition of a 2.5 inch Girder to the midway bend section of the arm. This addition allows for an increase of 2.5 inches in the arm's linear grappling range of motion. The previous CASCADE arm had a maximum range of 22 inches; adding the 2.5 inch girder and a dynamixel actuator for a 6th DOF will increase range to 28 inches. The Girder is made up of lightweight aluminum causing minimal

torsional effects on the joints due to weight and as well as provides an easy user interface for installation purposes.

Table 22: 2.5" Girder Addition Pros and Cons

Pros	Cons
2.5” of added extension	Smallest of possible extensions
Easy to install	Limited range capabilities for larger objects
Non invasive modification to arm	

2. 5” Girder

The second of two options augments the extension of the arm by adding a 5 inch girder to the midway bend of the robotic arm. The addition of the 5.0 inch girder and a dynamixel actuator for a 6th DOF will increase range to 30.5 inches. The Girder is made up of lightweight aluminum causing minimal torsional effects on the joints due to weight and as well as provides an easy user interface for installation purposes.

Table 23: 5.0" Girder Addition Pros and Cons

Pros	Cons
5” of added extension	Additional weight and torque on the arm without motor compensation
Easy to install	Poor maneuverability around objects at very close range
Non invasive modification to arm	

K. Mechanical Robotic Arm: Actuators

The system’s actuators are responsible for driving the mechanical arm to a desired location as well as supplying a constant force in order to grapple an object. Many factors have to be considered including cost and power in order to maximize efficiency and performance. The power requirements of the actuators are determined by their placement within the assembly; actuators closer to the base of the arm experience larger torsional forces due to added weight of motors and girders extending to the end effector. Under powered actuators have the potential to cause critical mission issues if they are unable to sustain the weight of the section while under operation (motor stall). Actuator stack ups require additional calculations to verify the capabilities of single motors within the assembly. The following studies outline two MX series actuators which provide both flexibility of the assembly as well as the necessary power to enable movement of the robotic arm.

1. Dynamixel MX-64T

The MX-64T one of the largest actuators of the MX series that CrustCrawler offers. The MX series actuators have a fully integrated DC motor, reduction gearhead, controller, driver and network on the module. The actuator also offers PID control, a 360 degree position control and high speed communication to the driver board (Dynamixel2USB). The MX-64T would serve as the actuator responsible for the movement of the arm to allow for the last DOF - the midway rotation of the robotic arm.

Table 24: Dynamixel MX-64T Actuator Pros and Cons

Pros	Cons
Use of Dynamixel actuator heritage hardware	Potential of running into same failure points
Added power ro actuate assembly	Smaller of two possible motors
Non invasive modification to the arm	Added weight and torque to mechanism

2. Dynamixel MX-106T

The MX-106T is the largest actuator of the MX series that CrustCrawler offers. The MX series actuators have a fully integrated DC motor, reduction gearhead, controller, driver and network on the module. The actuator also offers PID control, a 360 degree position control and high speed communication to the driver board (Dynamixel2USB). The MX-106T would serve as a replacement of the original MX-64T actuator at the base of the arm for a more robust and powerful actuator option to compensate for the additional weight and induced torque of the extension.

Table 25: Dynamixel MX-64T Actuator Pros and Cons

Pros	Cons
Use of Dynamixel actuator heritage hardware	Potential of running into same failure points
Added power ro actuate assembly	Higher cost
Non invasive modification to the arm	Added weight and torque to mechanism
Largest torque produced by MX actuators	

L. Ground and Test Support: Selection of Target Satellite Model

Sierra Nevada Corporation wished to expand on the efforts of CASCADE by switching focus from a small 3U cubesat to grabbable features present on commercial and military spacecraft. Due to how many satellites are in orbit and how different they are from one another, it is important to find similarities between them, and use that as the basis to build an algorithm that will be able to grab a large range of satellites. There are 804 active satellites in orbit in Low Earth Orbit (LEO)[35]. LEO is likely where the arm is likely to operate, as it contains the majority of the 1459 satellites in Earth orbit, in addition to the most debris. It is also relatively easy to access. To ease analysis, the group of satellites was searched, and the satellite series with the greatest number of active satellites in LEO were found. The satellites were the Iridium Series, the ORBCOMM FM series, the Yaogan and the Rodnik series, with 76, 40, 36, and 21 satellites currently in orbit, respectively. The Yaogan satellites are Chinese while Rodnik satellites are Russian. The foreign satellites are both military satellites, and therefore hard to get information on. The Iridium series, however, was easy to find information about.

M. Ground and Test Support: Selection of Grappling Feature

1. Solar Panel Joints

The idea of targeting solar panel joints comes from the fact that they are a major feature on every satellite. In addition, they are exceptionally consistent in between all types of satellites regardless of the country they originated from. This makes them an excellent choice for a project that hopes to be versatile and target a wide variety of satellites. In addition, the sharp contrast of the blue surface against the white paint will help to add additional veracity to the algorithms developed. Lastly, in many scenarios it is possible to guarantee an approach where the target area is illuminated by sunlight if the satellite is operational.

The major issue with targeting solar panel joints is that the solar panels themselves are extremely fragile, and even the smallest of scratches to the surface of the solar panels can cause the satellite to be left inoperable. On satellites to be decommissioned, however, the solar panel itself can become a valid target, as the solar panels are large, and are able to be approached by the arm while staying far away from the spacecraft. In both grabbing the solar panel joints on active spacecraft, and grabbing solar panels on soon-to-be-decommissioned spacecraft, recognition of the solar panel is crucial. Due to limited time and in an attempt to be as broad as possible with the range of capturable satellites, this project will be focusing on grabbing solar panel joints, though the recognition algorithm could be used with either joints or the panels themselves.

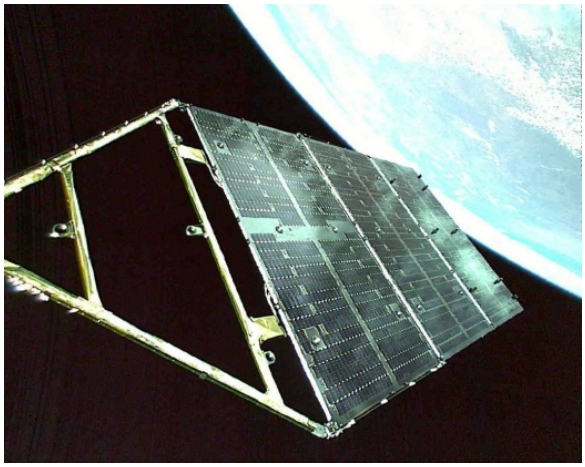

Figure 19: A sample image of satellite solar panels. [36]

Table 26: ProsCons for targeting Solar Panel Joints

Pros	Cons
Widespread Use in Operations	High Potential of Failure if Collision Occurs
High Visual Contrast	Structurally Flexible
Well Illuminated	Usually Off Center
Low Potential Collision of Other Features	

2. Bus Support Structure

The idea of targeting an onboard Bus Support System is that the satellite will be grappled by its most structurally secure point. Most of the time this feature is used to connect a smaller subsystem to the main body of the spacecraft. The length of the rods often mean that the arm has a chance to avoid collision with other features in the immediate area. Lastly the nature of the Bus Support means that they often are symmetrical such that approaching the satellite from any direction would allow for an approach. As such it would be possible to choose an angle that has both a lack of obstructions from the outside as well as a high level of illumination.

Unfortunately such systems do not always exist on satellites. Due to the nature of what a Bus Support is, targeting this system means that the arm becomes intended for Medium or Large Satellites rather than a wide variety. Most smaller satellites simply don't have a need for such a system and thus don't include it. Additionally the rods being round and metallic propose an issue when attempting to secure the satellite with the arm in its current configuration. The flat grabbing plates have a high likelihood to slip on the rod which means that the arm would require a fair amount of redesign to utilize.

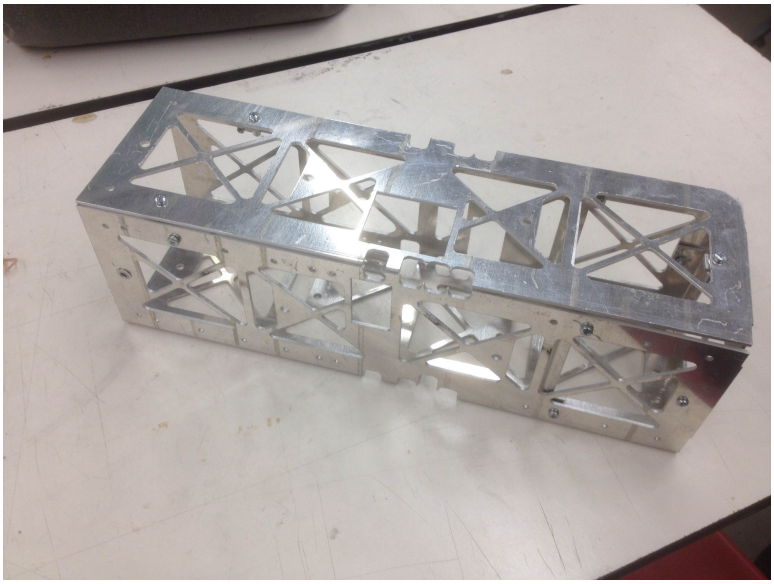

Figure 20: A sample image of a satellite Bus suport structure[37]

Table 27: ProsCons for targeting the Bus Support Structure

Pros	Cons
Structurally Secure	Not in Use on Smaller Satellites
Low Probability of Collision	Difficult to Secure in Current Configuration
Low Risk if Collision Occurs	
Multiple Approach Angles	

3. S-Band/GPS Antennas

The idea behind targeting one of the onboard antenna systems is that there is a sense of certainty about the targeted feature. While others will scale dramatically with the size of the satellite, often the antenna will be a similar size

regardless of the satellite. In addition the antenna often will be located in an area where it is unobstructed which can guarantee that an approach can be made without threat of collision with other features of the satellite.

The major downsides to targeting the antenna system is that if anything goes wrong with the execution of the command it is very possible to render a satellite inoperable as it will not be able to receive commands or transmit data. Targeting a GPS satellite then becomes a very risky operation that this arm might not be appropriate for. Additionally the antennas will almost always be oriented at Earth meaning that the grappling will have a probability for half of the orbit to result in a well lit target.

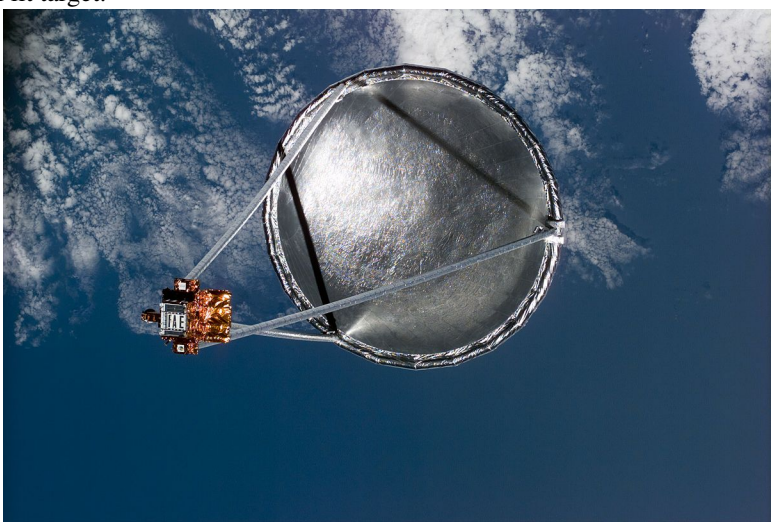

Figure 21: A sample image of a satellite antenna [38]

Table 28: ProsCons for targeting the Antennas

Pros	**Cons**
Consistent Size and Shape	High Risk if Collision Occurs
Low Risk of Collision with Other Features	Potentially Poorly Illuminated
Universally on Every Satellite	

4. Star Trackers

The idea of targeting star trackers is that they are a system that has a large amount of redundancy so the risk of rendering the satellite inoperable by damaging the grapple target is fairly low. In addition to the fact that often times Star Trackers have INS-type systems for stabilization as well, it is important to keep in mind that Star Trackers have to be well secured to the spacecraft as any amount of rattling during launch could easily dislodge the optics.

Star trackers are not available on many satellites, and it is difficult to determine their orientation before approach, though knowing the constellations the satellite is using for navigation and the orbit of the spacecraft it is possible to limit it to a few planes of existence. As such the spacecraft would have to be prepared to maneuver around the entire target satellite so identify them. Additionally Star Trackers tend to work much better when not pointed at the sun. This means that the star trackers will always be in a position where they are poorly illuminated making developing an algorithm much more difficult.

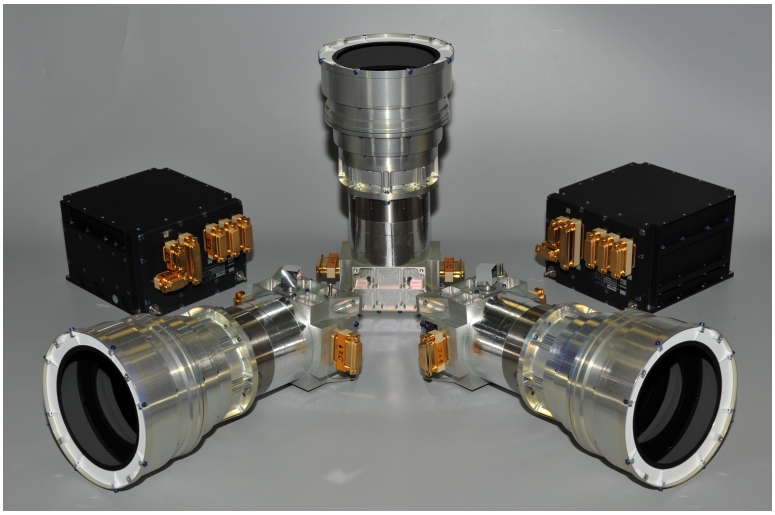

Figure 22: A sample image of a set of satellite star trackers[39]

Table 29: ProsCons for targeting Startrackers

Pros	Cons
Low Risk on Collision	Poorly Illuminated
Structurally Secure	Unpredictable Location
	Not on Every Satellite

5. ACS (Attitude Control System)

The idea of targeting the ACS is that the ACS is always near sections of the satellite that are very structurally secure. This is because of the relatively large amount of stress they exert on the satellite needing to be well distributed. Additionally when they are on a satellite they tend to be on every face. This means that regardless of the approach direction it is possible to find one exhaust port.

The reason that these are not they most likely target to be chosen is that they tend to either be flush with the satellite or non-existent at all. Either one poses a major problem, even if a satellite has a port that the arm can find, no protocol currently exists to open the arm to capture the satellite. This would require an upgrade to the base control code as well as upgrade to the arm's mechanical parts as well. Lastly in order to get any rotational effect out of the system the exhaust ports are required to function on a lever arm and will not be centered on the body. This doesn't affect the decisions for this project but, in the light of the customer's desire for this product in the future, it should be considered that grabbing a satellite from a large lever arm makes retraction very difficult to control.

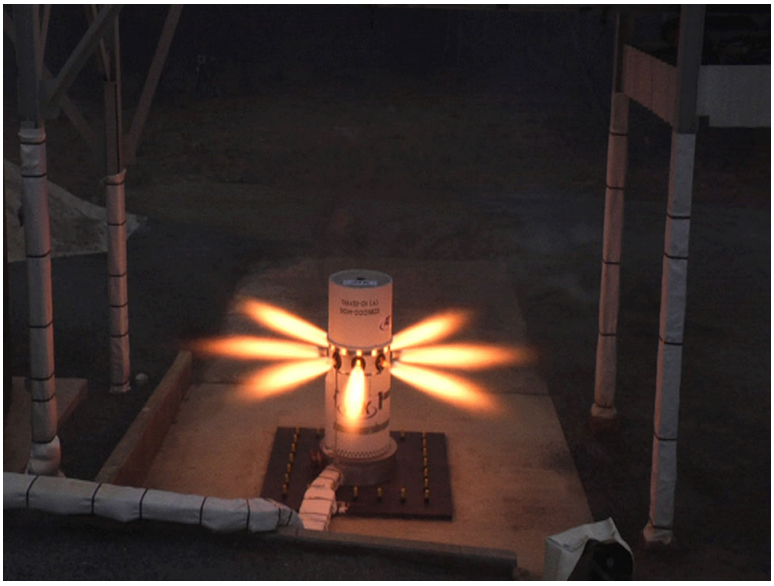

Figure 23: A sample image of the Orion attitude control system[40]

Table 30: ProsCons for targeting the ACS

Pros	Cons
Structurally Secure	Not Equipped to Grapple by Extension
Can Find a Good Approach Side	Difficult to Integrate in the Future
	Very Rare System for Satellites

N. Ground and Test Support: Simulated Satellite Size

1. Medium-Large Satellites

By simulating a satellite that is larger than 100 kgs, the arm will be geared towards potential refueling missions. This means that the algorithms and technology developed will be closer to flight ready. This will also allow for tests to reveal any unforeseen issues with the control sequences and geometry that a scale replica wouldn't identify. Additionally any software developed in the process of this project would not need to be scaled by the customer after delivery.

The challenge in a full scale model is that the arm would have to be rebuilt with larger structuring and stronger servos. This means that the project would have to entirely be worked from the beginning rather than building off of the work of the CASCADE group. This would result in a large increase in the cost of the project, the amount of work required, and the overall complexity of the system.

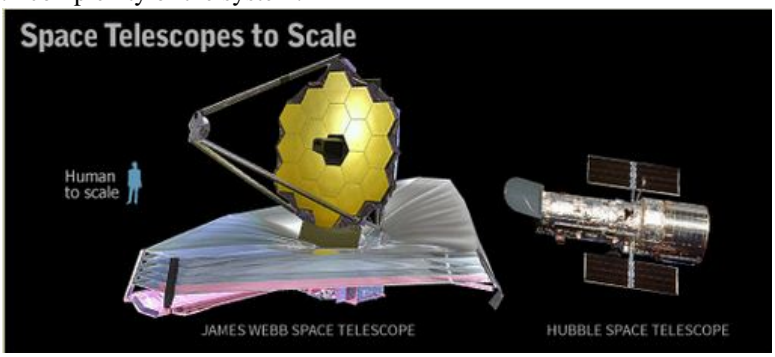

Figure 24: A comparison of the Hubble and James Watt Space telescopes (Large) compared to a human[41]

Table 31: ProsCons of using a Large/Medium Model

Pros	Cons
Less work Required by Customer	Higher Overall Cost
Finished System More Robust	More Time Required to Make
	Far More Complexity Involved in Redesign
	Requires impractical amount of space

2. Miniaturized Satellites

By simulating a satellite that is between 10 and 100 kgs in size or scaling down a larger satellite it greatly simplifies the amount of work that needs to be done during this project. By reducing the amount of time required in rebuilding the arm and controls algorithms, it is possible to increase the complexity of the image processing algorithms. This means that a wider variety of satellites could potentially be serviced by this product.

The drawback to implementing a system that is smaller in size is that the algorithms developed to help determine distance and size of the object will need to be rescaled. This is possible to do but will require extra work in developing the image processing code to easily allow it to scale to the appropriate satellite. Additionally the inherent issue of the existing system will need to be modified to make it strong enough to survive.

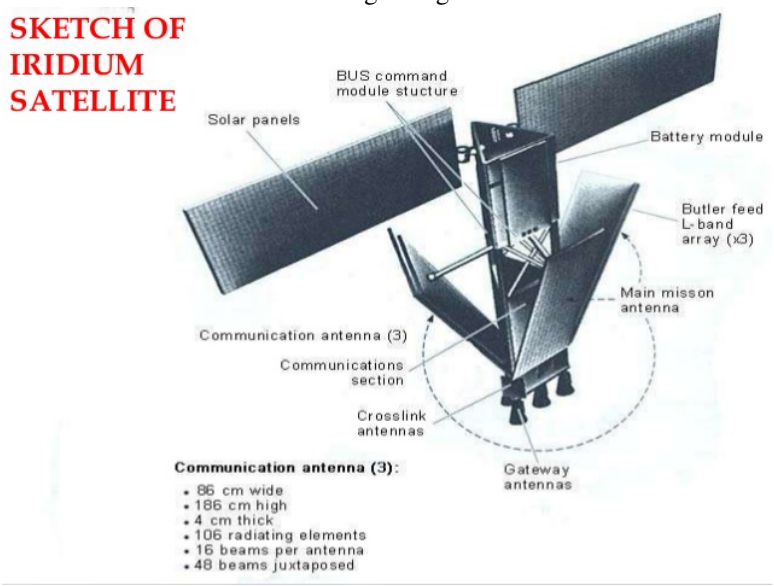

Figure 25: A diagram of the Iridium Series satellite detailing size[42]

Table 32: ProsCons of using Miniaturized Model

Pros	Cons
Less Cost for Mechanical Redesign	Requires Customer Refinement of the Software
Existing Interface can be Used	Arm Structural Health not as Robust

V. Trade Study Process and Results

A. Visual Processing: Imaging Hardware

1. Trade Study Criteria

Cost: The cost metric is defined by $100 for each ranking. The highest ranking is $0-$100 and the lowest ranking is $901+. This is an important metric since KESSLER's budget is only $5000.

User Documentation: User documentation refers to the amount of online support that can easily be found. Ideally for the hardware that is chosen there is a lot of support so that any issues that are encountered by the team can be quickly solved.

Picture Quality: Picture quality is also an important criteria because any camera that is chosen needs to be able to define a feature on the satellite. The pixel dimensions shown in the trade study table below are the most common pixel dimensions for the types of sensors that were researched for the trade study.
Supporting Software: This metric is simply categorized by the number of options of software that is available for each sensor. Later there is a trade study for the specific types of software that are compatible with the sensors.

2. Trade Study Metric Definitions

Table 33: Imaging hardware trade metrics definitions

	10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Cost	$0-100	$101-200	$201-300	$301-400	$401-500	$501-600	$601-700	$701-800	$801-900	$901+
User Documentation	Substantial documentation			A decent amount of documentation			Little documentation			No documentation
Picture Quality	1280x720 pixels			640x480 pixels			320-240 pixels			160x120 pixels
Supporting Software	Multiple software options				One software option					Discontinued software option

3. Trade Study

Table 34: Trade study results for imaging hardware.

	Weight (%)	**Microsoft Kinect**	**Orbbec Astra S**	**Intel RealSense SR300**
Cost	10	10	9	10
User Documentation	30	10	7	4
Picture Quality	30	7	7	10
Supporting Software	30	10	10	1
Weighted Total	100	**9.1**	**8.1**	**5.5**

B. Visual Processing: Secondary Visual Imaging Hardware

1. Trade Study Criteria

Resolution: The concern with having a secondary camera is to reduce the total error and provide a second measuring metric for our algorithm. It is defined by the Kinect resolution as the standard to meet, 640 x 480 pixels.
Microcontroller Ready Status: This metric was designed to help weed out camera devices that would require custom boards and additional breakout circuit design. This reasoning is because the goal is to be able to develop software with supporting hardware that will identify features. Thus the hardware should be as generic as possible for the case of scaling the project up or down.
Dimension and Weight: These metrics can be assigned for the same reason. If the device works flawlessly but hinders the capabilities of the arm, it isn't the proper choice. To reduce error the camera should influence the mechanics of the arm as little as possible. Thus the metrics of volume and mass were included.
Cost: Due to the budge of KESSLER and magnitude of cost that a camera can breach, a cost metric had to be included. The metric is between 0 and 100 dollars broken up by standard cost per unit of cameras for robotic implementation.
Power Requirements: The final metric is of power required to operate each device. The more complicated the power system the more likely the device will fail or cause surges for the arm. Standard voltages of 2.7, 3.3, 5, and 12 V were used to compare power strain on each device.

2. Trade Study Metrics

Table 35: Secondary Visual Imaging Trade Study Metric Table

	5	4	3	2	1
Resolution	>640 x 480		640 x 480		<640 x 480
Microcontroller compatibility	Raspberry Pi/ Arduino		Generic		Breakout Board Design Required
Dimension	<350mm^{2}		650 mm^{2}		>1300 mm^{2}
Weight	<=20g		>20g		>50g
Cost	<$30		$30-99		$100+
Power	2.7V	3.3V	5V		12V+

3. Trade Study

Table 36: Secondary Visual Imaging Hardware Trade Study

	Weight (%)	ArduCam CMOS OV7670*	Raspberry Pi Cam Module V2	CMOS Camera Module †	ArduCAM Mini 2MP OV2640	Pixy CMUcam5
Resolution	40	3	5	5	5	5
Microcontroller compatibility	20	5	5	1	5	5
Dimension	10	2	4	2	5	1
Weight	10	5	5	3	5	3
Cost	5	5	5	3	5	3
Power	15	4	-	2	4	3
Weighted Total		**3.75**	**4.15**	**3.15**	**4.85**	**4**

C. Visual Processing: Software

1. Trade Study Criteria

Documentation: Documentation describes the amount of information readily available about the software. This documentation generally gives information on how to use the software and its key features and functions and is very helpful when programming, as the programmer may be unfamiliar with certain aspects or functions of the software.
Availability of Visualization/Debugging Tools: The availability of visualization and debugging tools refers to tools that are available to help the programmer see what he or she is doing, as well as help identify possible bugs in the code that will need to be removed. Visualization also helps the programmer find faults in the code, because he or she will be able to see patterns and potential issues.
Availability of Library Functions/Toolboxes: The availability of library functions and toolboxes refers to built-in or add-on functions and tools that are available to help the programmer achieve his or her programming goals. These functions and tools are provided, allowing the programmer to use and usually alter functions that may already provide a certain need or may be used as a base for the programmer to achieve that goal.
Runtime: The runtime refers to the general, relative runtime of the various types of software being compared. Each type of software runs fast or slow compared to the others.
Difficulty of Use: Difficulty of use refers to how difficult the software is to use in general or to learn. This is relative, but each type of software has pros and cons that affect its difficulty.

2. Trade Study Metric Definitions

Table 37: Imaging software trade metrics definitions

	10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Documentation	Substantial docuentation			A decent amount of documentation			Little documentation			No documentation
Availability of Visualization/ Debugging Tools	Substantial number of tools			A decent amount of tools			Few tools			No tools
Availability of Library Functions/ Toolboxes	Substantial number of functions			A decent amount of functions			Few functions			No functions
Runtime	Fast			Somewhat fast			Somewhat slow			Slow
Difficulty of Use	Extremely easy to use			Somewhat easy to use			Somewhat difficult to use			Difficult to use

3. Trade Study

Discuss the results of the tradestudy

Table 38: Trade study results for imaging software.

	Weight (%)	**MATLAB**	**C++ with OpenCv**
Documentation	20	10	4
Visualization/ Debugging Tools	30	10	3
Availability of Library Functions/ Toolboxes	30	10	8
Runtime	5	3	9
Difficulty of use	15	7	3
Weighted Total	100	**9.2**	**5**

D. Control Subsystem: Software Integration Platform

The integration platform serves as an interface between the hardware and the core software elements. This is a valuable construction as it adds a layer of abstraction to the interface between the hardware and the main software. This way, the control and image processing software do not have to depend directly on the hardware selected for those purposes.

1. Trade Study Criteria

Legacy code: This shows how much of the previous work had been done in the given language. First, building new functions and modules from the ground up would take a considerable amount of time, so utilizing as much legacy code as possible would optimize efficiency. Second, changing legacy code from one platform (or language) to another would also be time-consuming and the team would like to avoid doing so, except for cases where the new implementation is needed in order to fulfilling time or memory requirements (i.e. better performance).
Visual hardware interface: This represents how easily the vision sensors' data can interface with the platform.
Control hardware interface: This represents how easily the thermal, servos, and other sensor data can interface with the platform.
Industry Availability: This metric is used to gauge how easy a customer hand off would be. Software that is written in a language which is not readily available in industry, or not familiar to the customer, can will lead to barriers in operations and further development by the customer.
User documentation: The idea of this requirement is to gauge how much documentation exists for tools and libraries in the chosen platform. If there is a large online community that uses the platform, online documentation can help the team develop code much faster.

Team Prior knowledge: This represents how knowledgeable the software team is with the platform. Using a platform that everyone knows well will save a significant amount of time and will lead to fewer errors during development.

The legacy code metric will be weighted the heaviest because of the large amount of time it would save to stay with existing code. Team prior knowledge and user documentation metrics will be weighted the lightest because although there may be a learning curve, it can be assumed that the team can pick up a new language relatively quickly. While

2. Trade Study Metric Definitions

Table 39: Trade Study Metrics Defined

	10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Legacy code	All code is in this language					About half of the inherited code is in this language				No inherited code is in this language
Visual hardware interface	All visual hardware interfaces easily with language					Some, but not all, visual sensors interface with language				No visual sensors interface with language, or there are time consuming obstacles
Control hardware interface	All control sensors interface easily with language					Some, but not all, control sensors interface with language				No control sensors interface with language, or there are time consuming obstacles
Industry availability	Software platform is easily attainable in industry					Software platform is available, but not common, in industry				Software platform is very hard or impossible to attain in industry
User documentation	Large online community/ documentation					General functions documented/in-depth questions not addressed				Independent code platform with little to no third party documentation
Team prior knowledge	All software members are comfortable with platform					Most software members have seen platform, but no extensive experience				Most software members have little to no experience. Large learning curve

3. Trade Study

Table 40 compares MATLAB and C/C++ with weighted values. Each metric score was multiplied by it's respective weighting and summed to receive a final score for the programming language.

Table 40: Trade Study Values

	Weight	**MATLAB**	**C/C++**
Legacy code	30%	10	1
Visual hardware interface	20%	10	10
Control hardware interface	20%	10	10
Industry availability	15%	9	10
User documentation	10%	10	10
Team prior knowledge	5%	10	9
Weighted Total	**100%**	**9.85**	**7.25**

As shown, MATLAB received a final score of 9.85/10 and C/C++ received a score of 7.25/10 in this trade study. Most metrics scored similarly. Because of their popularity and large online followeings, they both interface well with many different software and hardware components, and team members are familiar with both. The main difference comes from the "Legacy Code" metric. All previous code has been developed in MATLAB, so keeping a platform of MATLAB would save a significant amount of time and energy. There are no significant barriers that would prevent the implementation of either of these coding languages, so it can be concluded that MATLAB is the best option.

E. Control Subsystem: Controls Software

The control software has the primary task of receiving feature location data from the visual processing software, integrating that with knowledge of the current arm location, and creating a series of commands that will make the arm successfully move to and grab that feature. CASCADE implemented a suite of functions in MATLAB that commanded the arm to move to a specified location and grab. However, these functions did not contain the complete set of functions that are necessary for moving at angles that are not in the plane of the arm's base. The task of this component is to extend CASCADE's functionality into the more general case of orientation defined by the present project.

1. Trade Study Criteria

Legacy code: This represents how much existing code fulfills the function of this subsystem. This is a particularly valuable consideration as reusing existing code will save time that could be used to create code that will help fulfill requirements.
Team knowledge: This represents how well the team is already acquainted with the language. Time spent learning a new language takes away from productive development time, so capitalizing on existing proficiencies is a boon.
Matrix optimization: Matrix operations are a fundamental part of this system. The language chosen must have the capability to efficiently solve systems where matrices are multiplied together. The algorithms that the control system is based on use these matrix operations, and so this is also an important metric to judge these platforms.
Documentation and support: Some languages, like C, have large and active online communities that may be able to address problems that we may face, and therefore score well on this criterion. Languages also differ in the quality of the documentation of functions that are available to the user. This is a useful quality to have because if we run into bugs that have fixes published online, it is a simple matter to implement these fixes without taking significant time away from useful matters.
Industry availability: This metric shows how easily the customer can use the final product. For instance, C++ is widespread and distributed freely, while other languages like LabVIEW require expensive licenses or specialized developers to maintain. It is weighted low because all these languages were explicitly cleared as okay for use by the customer. Their relative ability to fulfill this requirement are therefore not very important.

2. Trade Study Metric Definitions

Table 41: Control software metrics

	10	9	8	7	6	5	4	3	2	1
Legacy code	Most code is in this language					About half of the inherited code is in this language				No inherited code is in this language
Team knowledge	All software members are comfortable with language					Most software members have seen platform, but no extensive experience				Most software members have little to no experience. Large learning curve
Matrix optimization	Language is built for efficient matrix operations					Language has matrix mathematics support available				Language has no support for matrices, we build our own
Documentation and support	Large online community, well documented functions					General functions documented/ in-depth questions not addressed				Independent code platform with little to no third party documentation
Industry Availability	Software is easily attainable in industry					Software platform is available, but not common, in industry				Software platform is very hard or impossible to attain in industry

3. Trade Study

Table 42: Control software trade results

	Weight	MATLAB	LabVIEW	C++
Legacy code	40%	10	1	1
Team knowledge	20%	10	7	8
Matrix optimization	20%	10	8	7
Documentation	15%	10	8	10
Industry availability	5%	9	9	10
Weighted Total	**100%**	**9.95**	**5.05**	**5.4**

This trade study shows MATLAB as the clear winner of this trade study. The only metric that it scored less than ideal on was the customer availability due to its proprietary nature and licensing requirements. Meanwhile, on the highest weighted metric (having code already available), it scored full points while neither of the other options have any existing code and score the lowest possible value.

F. Control Subsystem: End Effector Thermal Monitoring

The next trade study compares different solutions for servos that have been known to overheat on the arm. After 4 minutes of constant actuation (as the legacy code currently commands), the servos overheat. This can be solved in multiple ways: either through temperature sensors or by placing a time restriction in the control code. With temperature sensors, the sensors would send data back to the control code through a signal conditioning board or a microcontroller, and once the temperature passed a temperature threshold, the control code would command the servos off. With the option of just placing a time restriction in control code, once the timer hit a specified time, servos would be commanded off. This would eliminate the need for additional hardware and complexity. The following trade study compares advantages and disadvantages of three different thermo sensors and the implementation of a software time restriction. The metrics used to compare are size and weight, range, accuracy, interface, and cost.

1. Trade Study Criteria

Size & weight: This criteria specifies how easy it would be for the sensor to be attached to the end-effector servos. Since we do not want the sensor to affect the accuracy of the end-effector via drooping effects, size & weight are important factors to consider. This criteria is given a weight of 0.3 because it affects multiple critical project elements by affecting the accuracy and structural integrity of the arm.
Range: This criteria specifies whether the sensor meets the range of the internal operating temperature of the end-effector servos from -5 to 80 degrees C with some factor of safety. This criteria is given a weight of 0.15 because it determines whether the sensor satisfies its intended purpose.
Accuracy: This criteria specifies how accurate the temperature sensor is and whether this accuracy would affect the integrity of the failsafe. This criteria is given a weight of 0.2 because it determines whether the sensor satisfies its intended purpose.
Interface: This criteria specifies how easy it would be to interface the sensor to the control system so that the system can stop actuation of the servos. This criteria is given a weight of 0.3 because it affects the complexity of the system.
Cost: This criteria specifies how costly the overheating solution is. The project budget is $5,000, so the lower the cost the better. This criteria is given a weight of 0.1 because it affects the budget, which is an overarching constraint of the project.

2. Trade Study Metric Definitions

Table 43: End-effector Thermal Monitoring

	10	**5**	**1**
Interface	Easy to import data to chosen base platform	Need data acquisition device before importing to chosen base platform	Too complex
Size, Weight, Wiring	Size, weight, wiring are negligible	Size, weight, wiring cause substantial drooping of robotic arm	Can't put this on the robotic arm
Accuracy	Accurate within 1 degrees C	Accuracy is crude, but can still do the job	Accuracy is not sufficient
Range	Can detect temperature within the operating range of the servo and an extra 20+ degrees C	Can detect the temperature within the operating range of the servo	Temperature range doesn't extend to overheating range
Cost	$0-$25	$100-$125	$250

3. Trade Study

Table 44: Servo Overheat Solution Trade Study

	Weight	**Voltage Differential Temperature Sensor**	**Infrared Thermopile Temp Sensor**	**Thermocouple Breakout**	**Software Prevention**
Interface	30%	5	5	2	10
Size & Weight	25%	9	9	8	10
Accuracy	20%	8	10	8	10
Range	15%	5	5	10	10
Cost	10%	10	10	10	10
Weighted Total	**100%**	**7.1**	**7.5**	**7.6**	**10**

As shown above, the chosen design for the end-effector thermal monitoring subsystem is the is through software prevention. Software prevention received a score of 10 in each criteria because it does not carry physical size or weight, it does solves the problem by attacking the root cause of the problem, there is no need to import data from hardware to control software, and it does not affect our budget. The thermocouple breakout would be too complex for the subsystem because we need to purchase and integrate a 7inch long thermocouple to the end-effector. The infrared sensor has a score of 9 for size and weight, but a score of 5 for interfacing because we would need a data acquisition device to relay the data to the control software. The voltage differential temperature sensor got great scores as well, but will also need a data acquisition device to relay the data to the control software.

G. Mechanical Robotic Arm: Robotic Claw

1. Trade Study Criteria

Cost: Monetary cost of the claw.
User Documentation: User documentation is a method by which to better understand both the electrical and mechanical characteristics of the subsystem. Documentation can be provided by both the manufacturer or previous research conducted on the system.
Range of Extension: Outlines the maximum linear distance between grappling plates.
Motor Interface: The ease of use integrating CrustCrawler motors to robotic arm.
Contact Surface Area: The amount of surface area the grappling clamps contact on object.

2. Trade Study Metric Definition

Table 45: Robotic Claw Trade Study Metric Definitions

	5	**4**	**3**	**2**	**1**
Cost	$0 - 24	$25 - 49	$50 - 74	$75 - 99	$100+
User Documentation	4+ sources of manufacturer and heritage documentation.	3 sources	2 sources	1 source	No documentation available.
Range of Extension	Ability to grapple objects with cross sectional area of 9"+	Ability to grapple objects with cross sectional area of 6 - 8.99"	Ability to grapple objects with cross sectional area of 3 - 5.99"	Ability to grapple objects with cross sectional area of 1 - 2.99"	Ability to grapple objects with cross sectional area of 1" or less
Motor Interface	Capable of interfacing with all CrustCralwer motors	Two lines of CrustCrawler motors	A single line of CrustCrawler motors	2+ models of CrustCrawler motors	A single model of CrustCrawler motors
Contact Surface Area	Entirety of claw contacts object	75% of claw contacts object	50% of claw contacts object	25% of claw contacts object	Claw is capable of pointwise contact

3. Robotic Claw Trade Study

Table 46: My caption

	Weight (%)	**SG Robotic Gripper**	**AX Dual Robotic Gripper**	**AX 2" in 1" Robotic Gripper**
Cost	10	1	3	2
User Documentation	25	3	5	3
Range of Extension	30	3	5	4
Motor Interface	10	5	3	1
Contact Surface Area	25	3	4	3
Weighted Total	100	**3.9**	**4.35**	**3.0**

After analyzing the robotic claw trade study, it is determined that the AX Dual Actuator Gripper is the most ideal for both grappling a wide array of different sized objects with its range of motion and the modularity of two actuators sustaining torque on the seperate claws of the gripper. It also serves as heritage hardware from last year's CASCADE team and is proven to be both user friendly and reliable.

H. Mechanical Robotic Arm: Girder Length

1. Trade Study Criteria

Cost: Monetary cost of girder.
Range: Outlines the additional linear and spherical range of the robotic arm added due to the extension.
Weight: Weight of the girder.

2. Trade Study Metric Definitions

Table 47: Girder Length Trade Study Metric Definitions

	5	4	3	2	1
Cost Range	$0 - 24	$25-39	$40-59	$60-99	$100+
Weight	0 - 2 oz.	3 - 5 oz.	6 - 10 oz.	11 - 20 oz.	21+ oz.
Range	30"	28"	26"	24"	22"

3. Trade Study

The Girder trade study below shows that the 5" Girder is the best option going forward. Range is weighted more heavily than Cost and Weight since the performance of the arm is vital to mission success. The weight and cost of the Girders are very small when compared to the project budget and actuator capabilities, respectively. The 5" girder provides an added 2.5" of range with relatively low cost and weight tradeoff, so it makes sense to choose this as the best option going forward.

Table 48: Girder Trade Study

	Weight (%)	**2.5" Girder**	**5.0" Girder**
Cost	30	5	4
Range	40	4	5
Weight	20	5	5
Weighted Total	100	**4.1**	**4.2**

I. Mechanical Robotic Arm: Actuators

1. Trade Study Criteria

Cost: Monetary cost of actuator.
Performance: Outlines the performance metrics of the actuator including supplied torque of the system.
Weight: Weight of the actuator.
Criteria 4: Add a description here (1-2 sentences)

2. Trade Study Metric Definitions

Table 49: Actuator Trade Study Metric Definition

	5	4	3	2	1
Cost	$0-199	$200-399	$400-599	$600-799	$800-999
Performance	No load speed of 60 RPM or greater and stall torque of 6 N*m or greater	No load speed of 50-59 RPM and stall torque of 5-5.9 N*m	No load speed of 40-49 RPM and stall torque of 4-4.9 N*m	No load speed of 30-39 RPM and stall torque of 3-3.9 N*m	No load speed of 20-29 RPM and stall torque of 2-2.9 N*m
Weight	50g or more	100g or less	150g or less	200g or less	250g or less
Gear Ratio	150:1 or greater	100:1 - 149:1	75:1 - 99:1	50:1 - 74:1	25:1 - 49:1

3. Trade Study

After analyzing the actuator trade study, it is determined that the Dynamixel MX-106T actuator is the most ideal for both supplying the amount of torque necessary to rotate the arm from the midway bend given the addition of weight from the actuator itself as well as the girder extension. The MX-106T also serves as heritage hardware on the robotic arm mechanism therefore is well documented and can be reliably integrated.

Table 50: Actuator Trade Study

	Weight (%)	MX-64T Actuator	MX-106T Actuator
Cost	20	5	4
Performance	25	4	5
Weight	30	3	3
Gear Ratio	25	4	5
Weighted Total	100	**3.9**	**4.2**

J. Selected Design

This design choice was made by analyzing the trade study data for each components and choosing the best option for each. Below is a trade study of 3 arm augmentation options. This study was done in order to verify that our design decision is the most feasible. Table 14 shows the criteria of the trade study, while Table 15 shows the trade study and how each metric parameter weighs in. Option 1 is composed of a 2.5" Girder and an MX-64T addition while keeping other hardware as is. Option is the addition of a 5" Girder and an MX-64T while keeping other hardware as is. Option 3 is adding a 5" Girder and MX-64T while replacing the base Girder MX-64T actuator with an MX-106T model for higher power.

Table 51: Final Design Metrics

	5	4	3	2	1
Cost	$0-299	$300-599	$600-799	$800-999	$1000+
Maneuverability	Best	Great	Good	Adequate	Poor
Range	30"	28"	26"	24"	22"
Dynamixel Performance	10 Nm	8 Nm	6 Nm	4 Nm	2 Nm

Table 52: Final Design Trade Study

	Weight (%)	Option 1: 2.5" Girder	Option 2: 5" Girder	Option 3: 5" Girder w/ MX-106T
Cost	10	4	4	2
Maneuverability	20	4	5	5
Range	30	4	5	5
Dynamixel Performance	40	4	4	5
Weighted Total	100	**4**	**4.5**	**4.7**

The Trade study prioritizes Dynamixel Performance highly because it is a crucial part to mission success. Range, Maneuverability, and Cost are all sequentially weighted lower with cost being the least important parameter. Although option 3 is the most expensive option, it provides the best Range, Maneuverability, and Dynamixel performance. Mission success out-weighs monetary costs for this project. Below are pictures of the updated CASCADE arm with all modifications made.

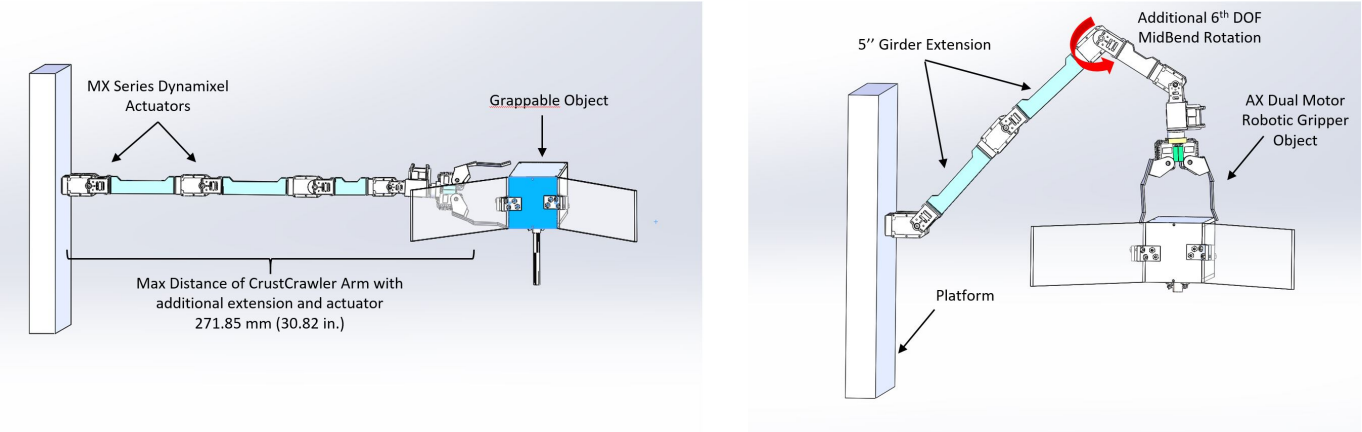

Figure 26: CASCADE Full Arm Extension and Overhead Grapple Configurations

K. Ground and Test Support: Simulated Satellite Size

1. Trade Study Criteria

There are four major categories that need to be considered when selecting the size of the satellite. The reason that this trade study needs to be done first is that selection of size of the satellite will drive the feature design selection. When looking at the size of satellites the two major sizes we will consider are both driven by the customers desire to utilize the arm for more than just cubesats.

Cost of Mechanical Redesign: This is representative of both the time and monetary cost of needing to redesign the physical system to allow the arm to reach all the necessary locations on the satellite. Given that the basic system isn't currently capable of manipulating in every axis, the highest level of success is considered a minimal cost rather than having no cost.

Use of Previous Work : The complexity of integrating new code and processes into the existing infrastructure. Given that new code will be added to the system it is expected that some work will need to be done. Where the scale quickly degrades is when the previous system fails to perform adequately for integration. Any time that a system has to be reworked because of compatibility it becomes a concern is this category.

Arm Structural Health: This is a gauge on how likely the arm is to fail not just during tests but after delivery. An arm that employs the use of heavier and more robust servos, controllers and power supplies has an overall greater health than one that utilizes servos operating near or beyond their capacity.

Customer Effort Required: The amount of effort that the customer will have to put in to maintain and manipulate the system for the purpose of utilizing it for other satellites. This system is judged by the success this project has in making the interface easy to change and scale to accompany a variety of satellites. A system that can successfully adapt to any satellite of choice would be considered a perfect success, while a system the that can only work with a singular satellite represented by the testbed designed by this project will be considered the lowest level.

2. Ground and Test Support: Trade Study Metric Definitions

	10	9	8	7	6	5	4	3	2	1
Cost of Mechanical Redesign	Expected outcome is to be done early and under budget including holdups			Expected outcome is to be on-time and under budget with low risk			Expected outcome is to be on-time and under budget but with a high risk to success			Expected outcome is to be late and over budget with low chance of success
Use of Previous Work	It all works perfect as is, nothing will need to be changed			One of the systems is not optimal but both are usable with little work			One or more systems are questionably usable to some degree			The previous project was done well but isn't pertinent to the work of this project
Arm Structural Health	There is no fear that the arm will fail in the next few years while run often			The arm is expected to last multiple years if it is well maintained and used carefully			The arm is expected to at best last a few months of careful usage before something breaks down			Turning it on more than once could result in the system breaking down
Customer Effort Required	A user only needs to supply the size of the satellite, orient the arm and sit back it is able to identify a feature regardless of size and color			The user is expected to interface with code in a basic level to help the arm find the satellite if multiple aspects of the feature change			The user is expected to interface with code at a deep level to teach the arm for new features but very similar ones still work			The user will be required to teach the robotic arm how to identify any feature not explicitly already on the satellite

Table 53: Trade Study for Satellite Size

	Weight (%)	Medium-Large	Mini or Scaled
Cost of Mechanical Redesign	30%	4	10
Use of Previous Work	25%	4	8
Arm Structural Health	15%	9	5
Customer Effort Required	30%	8	7
Weighted Total	100%	**5.95**	**7.85**

L. Ground and Test Support: Selection of Grappling Feature

1. Trade Study Criteria

Prominence in active satellites: The reason that active satellites were studied is that they tend to be representative of decommissioned ones that are still on orbit. Ones that are seen on every satellite are rated higher than ones that would need to be implemented going forward. This category is very important since tailoring the algorithm for feature that is common but harder to approach or harder to grab is more desired than looking for a perfect to grab feature that is not in use on any satellite. The algorithm ultimately should be able to grab most satellites, not just easy to grab ones.
Illumination of the sight: Another major factor is illumination of the sight. This is very important because of the requirement to use RGB sensors in design. Features that are oriented in a way where they are illuminated by the sun naturally are preferred over ones that can only be targeted if supplemental light is provided from the arm. If there the feature is not naturally illuminated, an artificial light would need to be used.
Structure Security: A feature that is naturally meant to be load bearing is prefered over one that isn't. This is because one that doesn't bear loads often could cause problems and could create debris. Ones that are core structural components are prefered.
Risk of collision with other feature: A concern of this project is the Risk that the arm faces in colliding with another feature on the target satellite. This in similar ways could cause undue damage to the spacecraft or debris if a collision occurred. Those with clear approach paths are greatly prefered over those with other large or fragile features nearby.
Risk of damage after improper approach: This is important to understand because targeting a feature that has a high risk of damaging the spacecraft if something goes wrong is less preferred to a feature that tends to be redundant or isn't likely to break on grapple.
Difficulty of orbital approach: If a feature is located on a single side or in a place that could be hard to locate the host satellite may have to maneuver multiple times in order to locate the proper feature which is not optimal. If a feature can be grappled from any directional approach it greatly adds to the robustness of the system as a whole and thus is prefered.
Complexity of object to recognize: the difficulty for the image processing software to identify the feature. This incorporates many different factors from the color contrast of the feature from the satellite to the uniqueness of the shape in comparison to other features. Something that stands out in visual color and shape from the rest of the satellite would be greatly prefered over a generically shaped white feature.

2. Trade Study Metric Definitions

	10	9	8	7	6	5	4	3	2	1
Prominence in active satellites	Every object desirable for targeting contains this feature and some contain duplicates			A majority of satellites on orbit contain this feature or contain something similar			Some objects contain the feature but a majority do not contain something similar			No satellites utilize the feature or something similar to target
Illumination of the sight	The feature sits in a location where it will likely be well lit by direct sunlight			The feature sits in a location where direct sunlight can possibly reach a majority of the orbit but not always direct			The feature sits in a shadow on the majority of the orbit and rarely sees any direct sunlight			The feature will require a series of searchlights in order to identify the object as it never sees direct sunlight
Structure Security	The feature is connected to a key load bearing structure on the spacecraft			The feature was not necessarily designed to bear a load, but it still is sturdy			The feature is fragile, and must be handled very carefully.			Never meant to be touched
Risk of collision with other feature	Isolated from other features			There is are minor objects in the way that complicate certain approaches.			The feature is accessible, but requires the arm to carefully plan and maneuver its rout.			A very bad game of operation
Risk of damage after improper approach	You could hit it with a bat and be fine			The component and the area around it is sturdy, but not invincible.			A minor mistake risks irreparably damaging the satellite, and major mistake would make it no longer operational			Anything but perfection would guarantee that the satellite is no longer operational
Difficulty of orbital approach	Distance is the only concern not attitude			The feature is accessible from most approaches, and it's possible to correct faulty approaches.			Careful planning is needed to approach the feature from orbit.			The satellite feature can only be approached from a very specific angle from orbit
Complexity of object to recognize	The computer can,recognize object using the simplest of algorithms with high success.			The computer requires a sophisticated algorithm to guarantee recognition.			Even with a sophisticated algorithm, there still remains a chance that the algorithm will be unable to identify the feature.			The object is an eldritch, non-euclidian nightmare

Discuss the results of the tradestudy

Table 54: Grappling Feature Trade Study

	Weight (%)	Solar Panel Joints	Bus Support Structure	Antennas	Star Tracker	ACS
Prominence in Active Satellites	25%	9	4	10	7	4
Illumination of the Feature	5%	10	7	7	4	10
Structure Security	10%	6	10	9	8	9
Structure Security	15%	10	9	8	5	6
Risk of Collision with Other Feature	15%	7	9	5	10	8
Risk of Damage After Improper Approach	10%	7	8	8	3	10
Complexity of Object to Recognize	20%	7	10	9	6	4
Weighted Total	100%	**8.00**	**7.85**	**8.30**	**6.50**	**6.30**

VI. Selection of Baseline Design

1. Visual Processing Subsystem: Selected Design

The visual hardware trade study results in the Microsoft Kinect sensor being the best sensor at the base of the arm. This option narrowly beat out the Orbbec Astra S sensor and overall is the better choice due to the fact that it was successfully used by CASCADE and has much more user documentation. The best way to verify the feasibility is to research an find someone who has used the Kinect sensor for similar applications. The Kinect sensor is a popular sensor to use for image tracking and there are many people who have documented and taken videos of image processing and tracking through the Kinect. This would prove that not only is the Kinect the best sensor to use, but it has worked in the past and will work for the applications of KESSLER.

The secondary visual imaging hardware trade study results suggest that the ArduCAM Mini 2 MP OV2640 is the best choice for claw-mounted secondary visual data taking. This is largely due to its outstanding rating of 4.85 out of 5 on the trade study. The only drawback of the device is its power requirements of 3.3V which is quite standard in most embedded system applications. In addition to that the device is designed to interface well with any microcontroller but is particularly pinned out and developed to work with Arduino. Arduino is a very MATLAB friendly platform and will allow for plenty of additional sensors if such a need arises. It is extremely testable hardware with extensive external

documentation and troubleshooting. From here forward we can verify feasibility by mounting pseudo-equipment of the same volume and mass in order verify the insignificant impact of the additional hardware on the arm. To verify that the equipment will work will require the hardware and testing using development kits and use of first hand user project documentation.

The visual processing software that is chosen based off of the trade study is MATLAB. MATLAB has much more documentation and debugging tools and C++ and it is much easier to use and more well known by the KESSLER team. The only downside to MATLAB is it has a slower run time in general than C++ but since timing is not an issue this is a relatively small issue and MATLAB is still the clear choice for software. The best way to verify the feasibility of MATLAB as the software for visual processing is to look at existing MATLAB documentation and projects. Since MATLAB is a widely used programming language with very good documentation, examples, and built in functions. Seeing proof that someone else has been able to use MATLAB for visual processing and identifying an object proves that MATLAB is suitable for KESSLER's visual processing applications.

2. Control Subsystem: Selected Design

MATLAB with its version of the Dynamixel SDK was found to be the best choice for the control software. Its excellent matrix manipulation strengths and especially the existing code base made it a very attractive option.

The visual processing team, through its own independent trade study, found MATLAB to be the best software for visual processing. That means the control software will have no interface issues with the visual software.

The software prevention code was chosen as the best option for solving the overheating problem with the end-effector servos. It's physical size and weight is non-existant, and it adds close to zero complexity to the system since it does not require hardware.

These together drove the adoption of MATLAB as the language for the base platform that will communicate information between the hardware and software. It has been shown from multiple trade studies that all visual and control software and hardware will interface well with MATLAB. MATLAB is also user friendly, the team is knowledgeable in the language, and it is easily available in industry. Therefore, any additional code that ties the visual and control sides of the project will be written in MATLAB. As one more advantage, MATLAB has effective libraries for communicating with data acquisition devices by serial protocols. MATLAB has libraries that enable serial communication, which is the standard protocol for many microcontrollers. This allows communication of data back and forth. If an Arduino microcontroller will be used in the future, MATLAB has particularly effective libraries designed specifically for communication with the Arduino.

3. Robotic Arm Subsystem: Selected Design

After evaluating all the different claw, actuator, and girder options for arm modifications it was decided that the best design option would be to add a 5" Girder with an MX-64T to the midsection of the arm while keeping the AX Dual Motor Robotic Gripper. On top of these additions, the existing MX-64T motor connected to the primary base Girder will be upgraded to the MX-108T actuator in order to account for the added weight. This design choice was made by analyzing the trade study data for each components and choosing the best option for each.

The 5" Girder will add the necessary range and maneuverability needed to grapple the object from the side faces rather than just the front (closest) face. The MX-64T provides 7.3 Nm of stall torque, which is enough to operate the CASCADE arm's mid-to-end section while still leaving room for extra weight to be added if needed. The AX Dual Motor Robotic Gripper was selected due to it's availability and performance; the claw is already part of the arm and is able to meet all of the project's performance requirements. Finally, the replacement of the existing MX-64T with an MX-106T will provide extra torque and power to the arm as it now has to operate a modified section with added weight. This selection was made as a precaution to ensure flawless motor performance during operation.

4. Ground and Test Support Subsystem: Selected Design

Ultimately the results of these trade studies reveal a clear choice forward when considering the size of the satellite. Going forward the KESSLER group will be choosing to scale down a full scale satellite to just a few kgs in weight. The major advantages of this decision is that the arm is already in a place where it should be adequate to reach any point on the satellite. The simplicity of not having to rebuild the entire project from the CASCADE team made this a clear decision.

Unfortunately when deciding on a grappling feature the trade study was far less helpful. Three results were nearly tied, being Solar Panel Joints, Bus Support Structures, and Antennas. Each has its own advantages but show that they could be optimal targets depending on the situation involved and all are in a similar range. As a result the KESSLER group will be exploring how to target and grapple all three with the first focus being on antennas. Beyond being slightly higher in the trade study it was also one that was suggested by Sierra Nevada Corporation as a possible target near the beginning of the project.

Figure 27 shows the summary of the baseline design for the KRESSLER Project with the trade flow diagram.

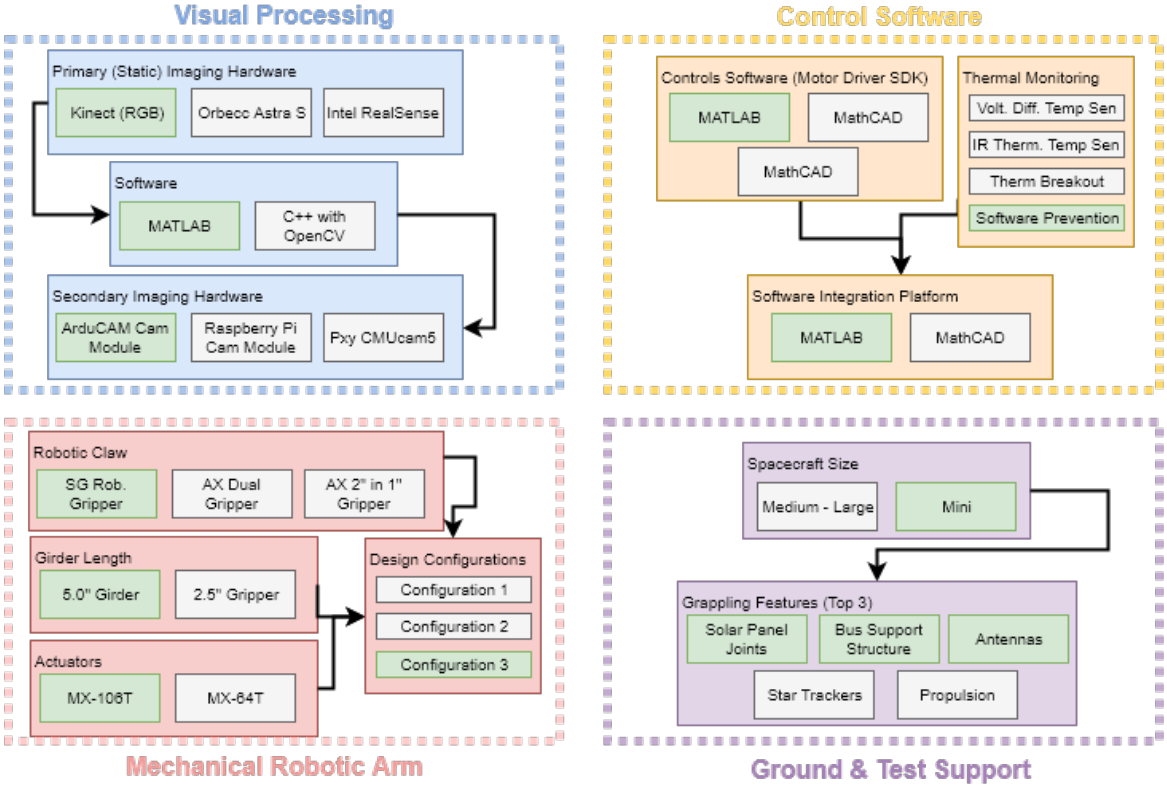

Figure 27: KESSLER Subsystems Trade Flow Diagrams - Selected Baseline Design

References

[1] David, Leonard. "Ugly Truth of Space Junk: Orbital Debris Problem to Triple by 2030", Retrieved September 10, 2017 from https://www.space.com/11607-space-junk-rising-orbital-debris-levels-2030.html

[2] O'Neil, Ian. " Orbital Spares: Iridium Already Replaced Destroyed Satellite", Retrieved September 30, 2017 from https://www.universetoday.com/25447/orbital-spares-iridium-already-replaced-destroyed-satellite/

[3] "Kinect", Retrieved September 25, 2017 from https://en.wikipedia.org/wiki/Kinect

[4] Stephanie Crawford. "How Microsoft Kinect Works", Retrieved September 25, 2017 from http://electronics.howstuffworks.com/microsoft-kinect2.htm

[5] "Installing the Kinect for Windows Sensor Support Package", Retrieved September 31, 2017 from https://www.mathworks.com/help/imaq/installing-the-kinect-for-windows-sensor-support-package.html

[6] "Kinect hardware", Retrieved October 1, 2017 from https://developer.microsoft.com/en-us/windows/kinect/hardware

[7] "Orbbec Astra S", Retrieved September 25, 2017 from http://shop-orbbec3d-com.3dcartstores.com/Orbbec-Astra-S_p_21.html

[8] "Orbbec Astra, Astra S Astra Pro", Retrieved September 28, 2017 from https://orbbec3d.com/product-astra/

[9] "Intel® RealSense™ Developer Kit (SR300)", Retrieved September 25, 2017 from https://click.intel.com/intelrealsense-developer-kit-featuring-sr300.html

[10] "Introducing the Intel® RealSense™ Camera SR300", Retrieved September 30, 2017 from https://software.intel.com/en-us/articles/introducing-the-intel-realsense-camera-sr300

[11] "Intel® RealSense SDK for Windows* (Discontinued)" Retrieved September 25, 2017 from https://software.intel.com/en-us/realsense-sdk-windows-eol

[12] "Computer Vision with MATLAB", Retrieved September 25, 2017 from https://www.mathworks.com/training-schedule/computer-vision-with-matlab?s_tid=srchtitle

[13] "Computer Vision Systems Toolbox", Retrieved September 25, 2017 from https://www.mathworks.com/products/computer-vision/features.html#object-detection-and-recognition

[14] Satya Mallick. "OpenCV (C++ vs Python) vs MATLAB for Computer Vision", Retrieved September 25, 2017 from https://www.learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/

[15] "OpenCV", Retrieved September 25, 2017 from http://opencv.org/

[16] "A Brief Description - C++ Information", Retrieved September 25, 2017 from http://www.cplusplus.com/info/description/

[17] "ArduCAM Mini Camera Module Shield w/ 2 MP OV2640 for Arduino", Retrieved September 30th, 2017 from http://www.robotshop.com/en/arducam-mini-camera-module-shield-2-mp-ov2640-arduino.html

[18] "Raspberry Pi Camera Module V2", Retrieved September 30th, 2017 from https://www.sparkfun.com/products/14028

[19] "Pixy CMUcam5 Image Sensor", Retrieved September 30th, 2017 from http://www.robotshop.com/en/pixy-cmucam5-image-sensor.html

[20] "ArduCam CMOS OV7670 Camera Module", Retrieved September 30th, 2017 from http://www.robotshop.com/en/arducam-cmos-ov7670-camera-module.html

[21] Leon. “Labview API", Retrieved September 28, 2017 from https://github.com/ROBOTIS-GIT/DynamixelSDK/wiki/5.7-LabVIEW

[22] Leon. “Matlab API", Retrieved September 28, 2017 from https://github.com/ROBOTIS-GIT/DynamixelSDK/wiki/5.6-MATLAB

[23] Leon. “C++ API", Retrieved September 28, 2017 from https://github.com/ROBOTIS-GIT/DynamixelSDK/wiki/5.2-CPP

[24] “Serial Interface (RS-232 and RS-485) and MATLAB.” MATLAB Serial, The MathWorks, Inc., www.mathworks.com/products/instrument/supported/serial.html.

[25] “Arduino Support from MATLAB.” Hardware Support, The MathWorks, Inc., www.mathworks.com/hardware-support/arduino-matlab.html.

[26] Conrad Sanderson and Ryan Curtin. “Armadillo: a template-based C++ library for linear algebra." Journal of Open Source Software, Vol. 1, pp. 26, 2016.

[27] Van Zijl, Johannes. "Space Debris Hit the International Space Station Causing Small Crack in Window", Retrieved September 10, 2017 from http://thescienceexplorer.com/technology/space-debris-hit-international-space-station-causing-small-crack-window

[28] Primack, Joel. "Debris and Future Space Activities", Retrieved September 12, 2017 from http://physics.ucsc.edu/cosmo/Mountbat.PDF

[29] NASA. "Restore-L Factsheet", Retrieved September 13, 2017 from https://www.nasa.gov/sites/default/files/atoms/files/restore_l_factsheet_062816_01.pdf

[30] Satellite Servicing Projects Division. "The Robotic Servicing Arm", Retrieved September 13, 2017 from https://sspd.gsfc.nasa.gov/robotic_servicing_arm.html

[31] Satellite Servicing Projects Division. "Restore-L Robotic Servicing Mission", Retrieved September 18, 2017 from https://sspd.gsfc.nasa.gov/restore-L.html

[32] Emergent Space Technologies. "Modeling & Simulation", Retrieved September 18, 2017 from http://www.emergentspace.com/capabilities/modeling--simulation/

[33] Satellite Servicing Projects Division. "NASA Satellite Servicing Evolution", Retrieved September 18, 2017 from http://images.spaceref.com/fiso/2017/011117_reed/Reed_1-11-17.pdf

[34] Satellite Servicing Projects Division. "Relative Navigation System", Retrieved September 18, 2017 from https://sspd.gsfc.nasa.gov/Relative_Navigation_System.html

[35] Union of Concerned Scientests, Retrieved September 20, 2017 from http://www.ucsusa.org/nuclear-weapons/space-weapons/satellite-database#.WcKrEMiGPt8

[36] Image of Solar Panels on Orbit, Retrieved October 1st, 2017 from https://www.intechopen.com/source/html/18984/media/image2.jpeg

[37] Image of BUS Support Structure, Retrieved October 1st, 2017 from http://usst.ca/wp-content/uploads/2014/12/IMG_2586.jpg

[38] Image of Inflatable Antenna on Orbit, Retrieved October 1st, 2017 from https://upload.wikimedia.org/wikipedia/commons/thumb/7/74/Inflatable_Antenna_Experiment.jpg/1200px-Inflatable_Antenna_Experiment.jpg

[39] Image of Star Trackers, Retrieved October 1st, 2017 from http://www.esa.int/var/esa/storage/images/esa_multimedia/images/2012/11/hydra_startracker/12111960-3-eng-GB/Hydra_startracker.jpg

[40] Image of Attitude Control System form Orion, Retrieved October 1st, 2017 from http://104.131.251.97/spacecraft/wp-content/uploads/sites/18/2015/09/4773040_orig.jpg

[41] Image Comparing Hubble, JWST and a Human, Retrieved October 1st, 2017 from http://1.bp.blogspot.com/-Kp6r0RdEez8/VZgxapY5AnI/AAAAAAAAACI/mxV8KXTrwKg/s640/Jwst%2Band%2Bhubble%2Bcomparison.JPG

[42] Diagram and Drawing of an Iridium Series Satellite, Retrieved October 1st, 2017 from https://image.slidesharecdn.com/presentation1-160621104245/95/iridium-satellite-system-9-638.jpg?cb=1466505804

Appendix

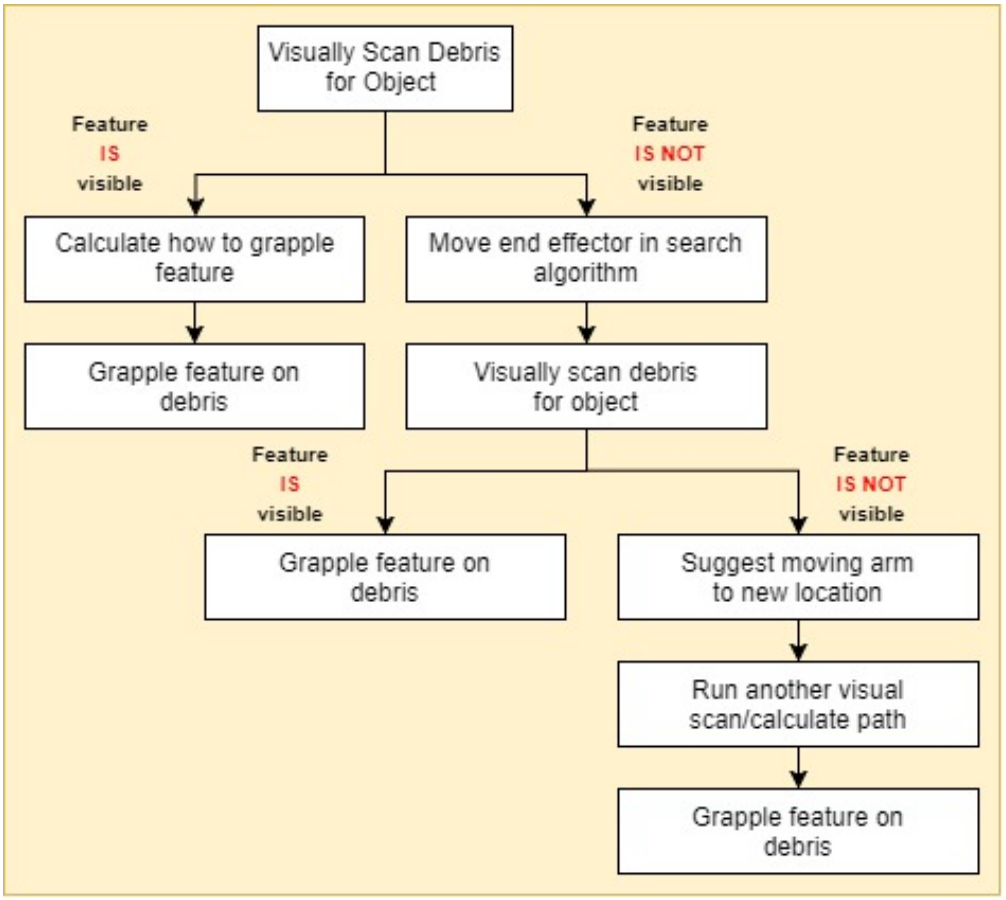

Figure 28: KESSLER Preliminary Software Logic Flow Diagram