

University of Colorado
Department of Aerospace Engineering Sciences
ASEN 4018

Conceptual Design Document - MEGACLAW
Mechanically Engineered Grappling Arm to Capture Litter and Atmospheric Waste

Monday 1st October, 2018

1. Information

Project Customers

William Jackson bill.jackson@sncorp.com 710-407-3101	TJ Sayer tj.sayer@sncorp.com 303-746-8962
---	--

Team Members

Luke Beasley (Manufacturing Lead) luke.beasley-1@colorado.edu 918-740-4425	Joseph Beightol (Safety Lead) jobe3924@colorado.edu 303-521-4048
Benjamin Elsaesser (Controls Lead) benjamin.elsaesser@colorado.edu 302-383-5771	Sheridan Godfrey (Test Engineer) sheridan.godfrey@colorado.edu 303-910-4912
Caleb Inglis (Project Manager) cain1127@colorado.edu 303-718-4566	Jack Isbill (Systems Engineer) jais8306@colorado.edu 303-945-0957
Andy Kain (Electronics Lead) david.kain@colorado.edu 859-552-0790	Cedric Leedy (Analysis Lead) cele7948@colorado.edu 303-319-9113
Chris Leighton (Financial Lead) chle0996@colorado.edu 720-833-1997	Daniel Mastick (Optics/Sensing Lead) daniel.mastick@colorado.edu 605-214-3527
Bailey Topp (Software Lead) bato9895@colorado.edu 610-952-3533	

Contents

1	Information	1
2	Project Description	4
2.1	Project Overview	4
2.2	Project Objectives	4
2.3	Concept of Operations	5
2.4	Functional Block Diagram	6
2.5	Functional Requirements	7
3	Design Requirements	7
4	Key Design Options Considered	9
4.1	Sensor Location	9
4.1.1	At End Effector	9
4.1.2	On Arm	10
4.1.3	At Base of the Arm	10
4.1.4	Offset from Base	10
4.1.5	At End Effector and Offset from Base	10
4.2	Sensor Hardware	11
4.2.1	Microsoft Kinect V2	11
4.2.2	Intel RealSense Depth Camera D435	11
4.2.3	Real Sense SR300	12
4.2.4	OPAL-P500	12
4.2.5	Snoopy V-Series VUX-IHA	13
4.3	Sensor Data Processing Software	13
4.3.1	Development	14
4.3.2	Operational	15
4.4	Control Software	16
4.4.1	Development	17
4.4.2	Operational	18
4.5	Hardware	19
4.5.1	Grappling Point Manufacturing	19
4.5.2	Grapple Point/Object Design	21
4.5.3	Arm Design	22
4.5.4	Design 1	24
4.5.5	Design 2	25
4.5.6	Design 3	26
4.5.7	End Effector	27
5	Trade Study Process and Results	29
5.1	Sensor Trade Studies	29
5.1.1	Sensor Location	29
5.1.2	Sensor Type	31
5.2	Software Trade Studies	33
5.2.1	Sensor Data Processing Software	33
5.2.2	Control Software	38
5.3	Hardware Trade Studies	41
5.3.1	Grapple Point Manufacture	41
5.3.2	Grapple Point/Object Design	42
5.3.3	Robotic Arm Design	44
5.3.4	End Effector Design	45

6	Selection of Baseline Design	46
6.1	Sensor Location and Hardware Baseline Choices	46
6.2	Software Baseline Choices	46
6.3	Hardware Baseline Choices	47

2. Project Description

2.1. Project Overview

As space is becoming more accessible, altitudes near Low Earth Orbit (LEO) are increasingly crowded with retired satellites, launch vehicle upper stages, and miscellaneous orbital debris. Currently, there are an estimated 13,000 objects of diameter greater than 10 [cm], 100,000 of diameter greater than 1 [cm], and tens of millions of objects with diameter less than 1 [cm] in aimless orbits throughout LEO.¹ Furthermore, in the coming years, there are plans to launch hundreds of CubeSats, as well as entire satellite constellations, into this environment.²

To mitigate this situation, Sierra Nevada Corporation (SNC), alongside the Ann and H.J. Smead Aerospace Engineering Sciences Department undergraduate senior projects team at the University of Colorado Boulder, shall develop a satellite-mounted robotic grapple-arm debris capture system. Success for this mission includes debris identification, capture and neutralization, with the purpose of either de-orbiting or salvaging said debris. As in previous iterations of this project, this team will develop the MEGACLAW to carry out capture and neutralization procedures. MEGACLAW shall use a sensor on a CrustCrawler robotic arm to identify both the target object and the arm's end-effector, and adjust in real-time to execute the arm's path to a defined capture point. This shall be achieved through the use of a closed-loop sensor feedback control software package. SNC will handle all structural, electronic, and thermal interfacing between the MEGACLAW and its bus to support this capturing process. The target object in this mission will have no rotational or translational motion relative to the arm, dimensions that allow full view of the debris from the arm's viewpoint, and a capture point designed for grapple tests.

Much credit is due to SNC's past two senior projects, CASCADE⁷ and KESSLER⁸ from years 2016-2018. CASCADE developed rigid body dynamics models in MATLAB, designed a robotic arm with five degrees of freedom (DOFs), and, using an external vision system, demonstrated success in capturing "active" CubeSats rotating about a single stable axis. KESSLER improved upon this design by developing a 7 DOF arm with an integrated vision system using a Kinect V2, capturing a scaled Iridium satellite by executing an open-loop arm path planning system. MEGACLAW will build upon these innovations to improve the vision system, software processing, and arm actuation efficiency in order to adjust the arm's path during capture execution using real-time feedback from the vision system.

2.2. Project Objectives

As discussed above, the motivation for this grapple arm design, to clear orbital debris, is of utmost importance to help protect future spacecrafts. While the previous group (KESSLER) developed a successful grapple mechanism and achieved their design requirements, the project operates under the simplifying assumption that the orbital debris was stationary relative to the grapple arm. As such, KESSLER implemented an open loop system to grapple the target object based on preliminary sensor data. Though the aforementioned design has its uses, the majority of orbital debris is characterized by some rotational and translational movement. As such, using an open-loop system is not optimal to serve the ultimate purpose of the problem statement.

In order to address the build-up of existing orbital debris, MEGACLAW will implement a closed-loop control system, in combination with a CrustCrawler robotic grapple arm, to track, approach, and capture a mock-up TBD satellite. The control system's closed-loop shall be dependent on sensor data collected by a TBD sensing system, from which the relative positions and orientations of the robotic arm's end-effector, as well as a given (TBD) grapple point, will be computed. Upon receiving the aforementioned state data from the sensing system, the control software package shall determine the most optimal path for the end effector to follow in order to reach the target's grapple point at a desired orientation and will command the arm to proceed along that path - until it receives updated positions from the sensor system (which shall collect new readings and update the control system at TBD rate) and modifies the path accordingly. Upon reaching the grapple point on the target, the end effector shall secure the target.

To more specifically define the scope of this project, two discrete areas are identified as crucial in measuring the project's success. Under each area, three measurable levels of success are set. If in one of these areas the project achieves, for example, level 3 success, but in the other area achieves only level 1 success, then the entire project will only be said to have reached level 1. The two areas for which success shall be quantified are as follows:

- The system's spatial sensing system, which shall operate in a closed-loop, such that (1) the onboard camera collects data at a frequency of TBD Hz, (2) identifies and tracks objects in its field of view (and at higher success levels, a special grapple point of predetermined TBD shape), and (3) output data to a control system that will determine an updated position for the end effector to move to in order to most efficiently rendezvous with the target.

- The grapple arm, which shall (1) receive input commands from control software based on positional data found, (2) actuate corresponding motion to approach object, and (3) ultimately grapple the object.

With these two discrete areas, levels of success were determined for each benchmark of the project. The resulting objectives and their levels of success are displayed below in Table 1.

	Closed Loop Sensing	Arm Actuation
Level 1 Success	Identify and report end-effector location at TBD rate within TBD cm in each direction	End-effector responds to given controlled input within given error margin, to be determined as a function of end-effector's effective grapple range (subset of total grapple range, set such that only contact is between end-effector and target grapple point)
Level 2 Success	Identify and report grapple location and end-effector location at TBD rate within TBD cm in each direction	End-effector contacts with grapple point on object, at TBD speed, with a predetermined orientation as dictated by sensor system input
Level 3 Success	Identify, predict and report grapple location and end-effector location at TBD rate within TBD cm in each direction	Successful grapple of target object, denoted by equal non-zero force on all contact points between end-effector[s] and target

Table 1. Success Levels

2.3. Concept of Operations

The task that MEGACLAW will help accomplish is shown in the Figure 1 CONOPS. A satellite will rendezvous with space debris, MEGACLAW will capture the space debris, and finally the client can either perform maintenance or de-orbit the debris. The capture phase of this task is the scope of the MEGACLAW project. It is assumed that the satellite has rendezvoused to a constant relative position and orientation to the space debris. The MEGACLAW project has not been tasked to perform operations after capturing the debris. In order to simplify the problem, the client has specified that the capture can be assumed to occur during a 30 minute eclipse in the orbit. This will require MEGACLAW to provide an external light source that will be known and provided to the closed loop software as compared to the changing natural lighting of the sun during orbit.



Figure 1. Big picture concept of operations

Within MEGACLAW's scope, the CONOPS is shown in Figure 2. The main task for this project is to develop a closed-loop algorithm to continuously compute a path for the CrustCrawler arm to follow to a given point on the target object to be grappled. The software flow diagram displays the basic logic of this algorithm. The sensors onboard will continuously detect the relative positions and orientations of the grapple point (GP) and arm end-effector (EE), and compute a path for the arm to follow. The arm will then capture the debris at the target grapple point.

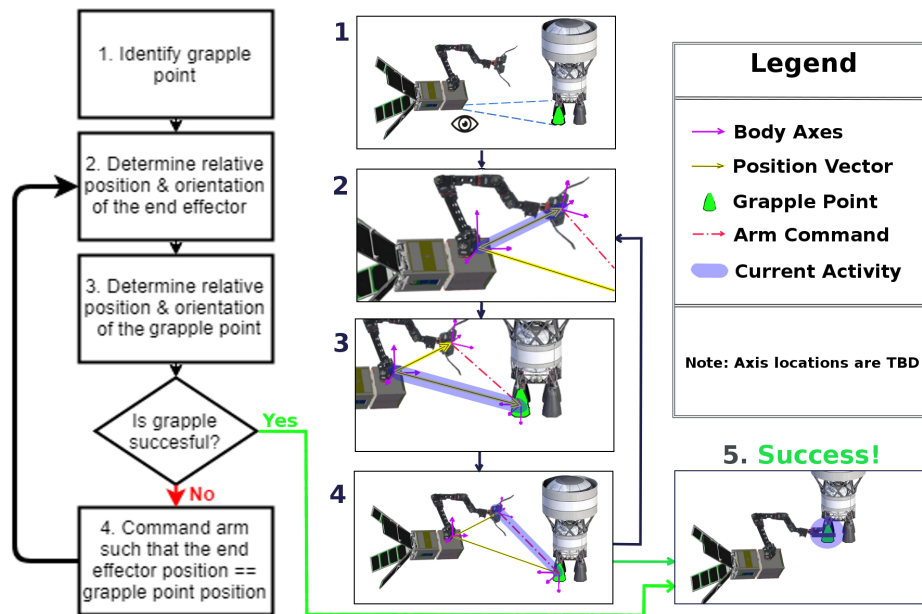


Figure 2. Concept of Operations

2.4. Functional Block Diagram

The FBD (Functional Block Diagram) shown in Figure 3 shows the functionality of the MEGACLAW system. The GSE hardware will include mounts for the CrustCrawler arm, target object, external lighting, and sensor packages. The arm and target object mounts will remain stationary. External lighting will provide a constant, known light source for the sensors. Sensors will send position and orientation data to the DAQ, which will communicate reference data to the CPU. Here, the closed-loop software will process said data, and compute a valid path for the arm to take to the target grapple point. Commands will be sent from the CPU to the DAQ. The DAQ will communicate these commands to the arm, which will move according to said commands until it has reached the target point, at which point it will capture the object. The closed loop software will complete this cycle at a TBD frequency to provide a continuously updated path to the arm, in order to improve upon the accuracy and flexibility of an open-loop control system.

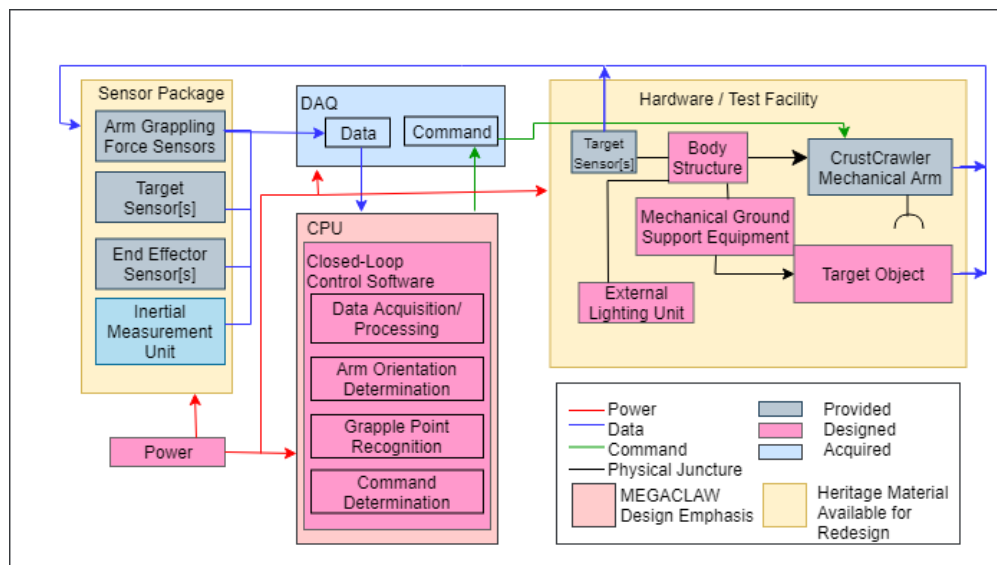


Figure 3. Functional Block Diagram

2.5. Functional Requirements

1	Grappling arm shall receive commands from the closed loop algorithm.
2	Grappling arm shall follow continuously updated commands given by closed-loop algorithm.
3	Control software package shall use feedback data to compute relative position points for end-effector pathing.
4	Ground support equipment shall be redesigned to minimize movement due to torque from the arm's actuation.
5	Grappling target used in system GSE shall have no translational or rotational motion. However, the control software shall be written such that these motions can be accounted for in the future.
6	All systems shall be operational in a 1G testing environment.
7	End-effector shall have ability to release object upon command.
8	Grappling arm shall avoid contact between debris and all of its components excluding end-effector.
9	Grappling arm shall avoid contact with host vehicle.

3. Design Requirements

FR 1 The grappling arm shall receive commands from the closed-loop algorithm.

Motivation: In the process of grappling, the arm needs to communicate with the software in order to command the actuators to move the arm.

DR 1.1 The entire grappling arm control loop will be designed to operate above a rate of 2π Hz (Value is TBR)

- Motivation: The control loop frequency is a parameter to be tuned when optimizing the control system. The value given here is based on the frequency of an object rotating at one revolution per second. This value will be revised in future documents once a good dynamics model has been developed that can be analyzed to determine the appropriate frequency.

FR 2 Grappling arm shall follow continuously updated commands given by closed-loop algorithm.

Motivation: The arm needs to be designed to be autonomous such that the arm will not receive manual commands.

DR 2.1 The actuators shall operate at a rate equal to or faster than the reception rate defined in **DR 1.1**.

- Motivation: Minimize control system lag.

FR 3 Estimation software shall use feedback data to compute relative positions of the end effector and grapple point to the sensor frame.

Motivation: Closed loop control requires feedback data consisting of the state of the system that is being controlled.

DR 3.1 End-effector position shall be determined within a 4mm linear tolerance.

- Motivation: This tolerance comes from KESSLER's grapple point position determination tolerance. This is because MEGACLAW aims to achieve determination accuracy at least equal to that of KESSLER's.

DR 3.2 The grappling point position will be determined within 4mm tolerance.

- Motivation: This tolerance comes from KESSLER's grapple point position determination tolerance. This is because MEGACLAW aims to achieve determination accuracy at least equal to that of KESSLER's.

DR 3.3 The orientation of the end-effector relative to the sensor will be determined within a 5 degree tolerance.

- Motivation: This tolerance comes from KESSLER's grapple point orientation determination tolerance. This is because MEGACLAW aims to achieve determination accuracy at least equal to that of KESSLER's.

DR 3.4 The orientation of the grappling point relative to the sensor will be determined within a 5 degree tolerance.

- Motivation: This tolerance comes from KESSLER's grapple point orientation determination tolerance. This is because MEGACLAW aims to achieve determination accuracy at least equal to that of KESSLER's.

DR 3.5 The estimation software's computational rate shall be designed as $\frac{3}{4}$ of the total rate defined in **DR 1.1**.

- Motivation: The estimator is expected to be the most computationally intensive component of the software package, so it is allocated $\frac{3}{4}$ of the total computation time budget.

FR 4 Ground support equipment (GSE) shall be redesigned to minimize movement due to torque from the arm's actuation.

Motivation: In the previous years project, KESSLER, the torque due to arm actuation in 1G caused the ground apparatus to rotate at the base of where the arm was located, leading to error in measurement of the distance between the end effector and the grappling point.

NOTE: Design requirements pertaining to GSE are set as children of **FR 5**

FR 5 Grappling target used in system GSE shall have no translational or rotational motion. However, the control software shall be written such that these motions can be accounted for in the future.

Motivation: The customer, SNC, has defined that MEGACLAW shall not consider a moving grappling point for its scope, though control and pathing software shall allow for future iterations of the project to build upon MEGACLAW's code and include said motion.

DR 5.1 The tolerance of movement of the grappling arm's base shall be measured with respect to a stationary point.

- Motivation: This requirement shall ensure that there will be no additional grappling arm base position error due to reference position error, thus tightening the error budget of the project.

DR 5.2 The tolerance of movement of the grappling point shall be measured with respect to a stationary point.

- Motivation: The motivation of this design requirement is matched with that of **DR 5.1** for consistency.

FR 6 All systems shall be operational in a 1G testing environment.

Motivation: As the scope of this project is proof of concept, the requirement for the grappling arm to actuate in zero G is not necessary. The members of team MEGACLAW also do not have access to a zero-gravity environment.

DR 6.1 The system shall support its own weight in a 1G environment.

- Motivation: As the grappling arm shall be tested and verified in a 1G environment, it must be able to operate for at least the duration of one full test while supporting its own weight.

DR 6.2 Actuators shall be designed to move within ± 90 deg under 1G whilst other actuators are moving.

- Motivation:

DR 6.3 All programs shall be written with a gravitational constant of $g = 9.80665$ [m/s/s] (1G).

- Motivation: This sets a standard for all calculations and designs within the control software algorithm to satisfy **FR 6**.

DR 6.4 Dynamics models shall be developed assuming a 1G environment.

- Motivation: Assumptions in the dynamic models shall be relevant to the corresponding testing environment.

FR 7 End-effector shall have ability to grasp and release object upon command.

Motivation: Though the scope of this project only denotes that the grappling point shall be grappled, the system must be designed with the eventual option to de-orbiting the object of interest, inspiring the latter half of this functional requirement. This context was provided by the customer, SNC.

DR 7.1 End-effector shall apply a force that exceeds the product factor of safety (**TBD** N) combined with the force sensor noise ceiling, applied symmetrically to the target.

- Motivation: A symmetric force loading on the grappling point shall ensure that no torque is applied to grappling point, and that the gripping system shall be stable (in equilibrium).

DR 7.2 The grappling of the end-effector shall last at least 5 seconds.

- Motivation: This duration shall ensure that no part of the end effector slips from the grappling point or releases the grappling point contrary to command, ensuring static stability.

FR 8 The path determination software shall determine a series of points in space for the end effector to follow based on the relative positions of the end effector and grapple point.

Motivation: A closed loop control system requires a desired state as an input.

DR 8.1 Grappling arm path shall be determined such that only contact between is between the end-effector and the grappling point.

- Motivation: Contact between other points of the system could cause damage to the arm or the target object.

DR 8.2 The path determination software's computational rate shall be designed to be $\frac{1}{8}$ of the total rate defined in **DR 1.1**.

- Motivation: This metric is based upon a textbook reference on basic guidelines for controls systems, suggesting that the controls system shall operate less frequently than the sensing system by a factor of $\frac{1}{8}$.

FR 9 The control software shall generate commands for the actuators such that the end effector follows the series of points defined by the path determination software in **FR 8**.

Motivation: This requirement shall ensure that the end effector follows the most efficient and clear path to the grappling point while avoiding obstacles.

DR 9.1 The actuator commands shall be generated such that the end effector follows the defined series of points in space within a tolerance of 4mm.

- Motivation: This is a tolerance derived from a project of heritage, KESSLER, and shall become the minimum standard for MEGACLAW.

DR 9.2 The control software's computational rate shall be designed to be $\frac{1}{8}$ of the total rate defined in **DR 1.1**.

- Motivation: The motivation for this requirement matches that of **DR 8.2**.

4. Key Design Options Considered

4.1. Sensor Location

The sensor location determines the point at which data will be collected, and affects many aspects of the overall design. MEGACLAW's design choices lead to 5 possible locations for the sensing system: at the end effector, somewhere along the grappling arm, at the base of the arm, offset from the base, or a two-sensor setup - one at the end effector, and one offset from the base. These location options are depicted in Figure 4 below.

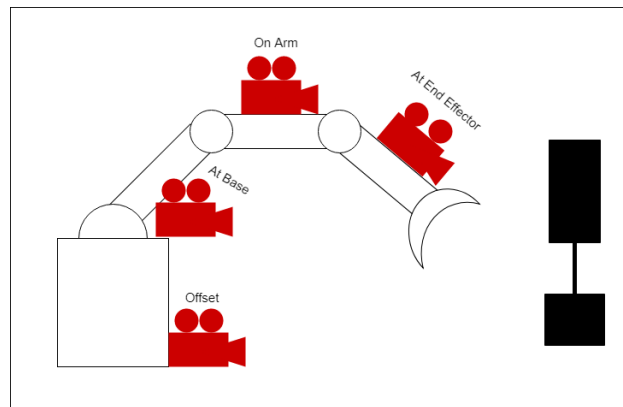


Figure 4. Sensor Locations

4.1.1. At End Effector

This scenario mounts the sensor on the grappling arm member before the gripping mechanism. Sensor location at the end effector results in a view that is only obstructed by the gripping mechanism. The cables necessary to operate the sensor will require being channeled against the arm in a manner that does not constrict movement. This location will require a complex design to mount the sensor to the arm. The design must hold the sensor stationary and prevent rattling during arm actuation. The torque required to move the arm will increase due to the added mass from the mount and sensor which could require stronger actuators. The sensing location will change position as the arm actuates towards the target object. The movement results in a stationary object appearing to have movement, and increases the likelihood of losing sight of the target object. Furthermore, this sensor location exposes the sensor to potential collisions with the target object. Damage to a critical project element would result in system failure.

Advantages	Disadvantages
Low visibility obstruction	Complicated mounting design
	Moving sensing location
	High risk to sensor

Table 2. Advantages and Disadvantages of Mounting Sensor at End-effector

4.1.2. On Arm

This scenario mounts the sensor on a member of the arm that is not directly before the gripping mechanism. The arm mount shares many benefits and disadvantages as the previous scenario. The sensing position moves with the actuation of the arm. The complicated system results in difficulty tracking the target object's motion. The main difference between the two options is the sensor's view is obstructed by the remaining arm members and the gripping mechanism.

Advantages	Disadvantages
Moderately low visibility obstruction; greater than at end-effector	Complicated mounting design
	Moving sensing location
	High risk to sensor

Table 3. Advantages and Disadvantages of Mounting Sensor on Grappling Arm

4.1.3. At Base of the Arm

This scenario mounts the sensor on the test bed (GSE) at the base of the robotic arm. Project KESSLER used a sensor mounted at the base. This mounting position results in a view that can be obstructed by the entire arm. The stationary sensor does not obstruct the movement of the arm. Mounting at the base would require a simple design to hold the sensor stationary.

Advantages	Disadvantages
Simple mounting design	Visibility obstructed by robotic arm
Stationary sensing location	
Low risk to sensor	

Table 4. Advantages and Disadvantages of Mounting Sensor at Base of Grappling Arm

4.1.4. Offset from Base

This scenario mounts the sensor on the test bed offset from the base of the robotic arm. Offsetting the sensor from the base puts distance between the sensor and the arm, reducing obstruction due to the arm. The location offset from the base shares many benefits with the location at the base. The simple mounting design will hold the sensor stationary throughout the grappling process.

Advantages	Disadvantages
Simple mounting design	Visibility obstructed by robotic arm, though less than base mounting location
Stationary sensing location	
Low risk to sensor	

Table 5. Advantages and Disadvantages of Mounting Sensor At Location Offset from Arm Base

4.1.5. At End Effector and Offset from Base

This scenario mounts two sensors to the robotic arm system. This design uses a medium range sensor offset from the base and a short range sensor at the end effector. This design would use the medium range sensor to determine the target object location and perform maneuvers to get the gripping mechanism closer. As the gripping mechanism

approaches the target object, it can use the short-range sensor to sense a more accurate position of the object. This scenario provides the most information on the location of the target object and end effector. This scenario also presents the most complicated mounting system, as it will use both designs described in sections 4.1.1 and 4.1.3.

Advantages	Disadvantages
Minimum visibility obstruction	Complicated mounting design
	Complicated control system

Table 6. Advantages and Disadvantages of Mounting Sensors at Both End-Effector and Offset from Arm Base

4.2. Sensor Hardware

The type of sensor used in this project is a pivotal decision, as it determines the ability to track both the end effector and the grapple point. The choice in sensor will also have an influence on the lighting requirements, so the following criteria were designed to help determine which sensor type is optimal for this application. The sensor types chosen for this application were all from one of the following types; Light Detection and Ranging (LiDAR), Red Green Blue (RGB), InfraRed (IR), and combined RGB+IR. All of these candidates were initially listed as suitable for the task of identifying the position and distance of a specific point, but each one comes with specific advantages and disadvantages. The decision was made to study specific sensors over studying sensor types due to the advantage of being able to study other projects using specific cameras. The other advantage is that, if specific sensors are examined, then the work of other projects using that specific sensor could be analyzed for the purpose of this trade study.

4.2.1. Microsoft Kinect V2

The Microsoft Kinect V2 is a depth camera originally designed as a sensing component for video games. The included sensors are a RGB camera, depth sensor, and an array of microphones. The depth sensor is an infrared laser projector communicating with a monochrome CMOS(Complementary metaloxidesemiconductor) sensor. The clear depth sensing range is between 0.5 and 4.5 meters. The depth resolution is 512 x 424 with a frame rate of 30 fps. The Microsoft Kinect has been discontinued, but the documentation supporting it is still available. Microsoft has a toolkit available that allows the Kinect to work on Windows 10.

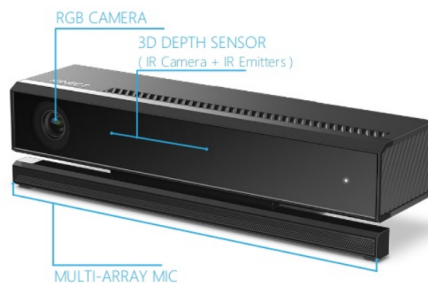


Figure 5. Microsoft Kinect V2

Advantages	Disadvantages
KESSLER heritage	Product discontinued
Sensor provided by SNC	Only Windows 10 compatible
Optimal range	

Table 7. Advantages and Disadvantages of Kinect V2

4.2.2. Intel RealSense Depth Camera D435

The Intel RealSense Depth Camera D435 calculates depth through the use of stereo vision. The included sensors are an RGB sensor and an infrared projector. The D435 camera has a minimum depth distance of 0.2 meters and a maximum range of 10 meters. The depth field of view is 86° x 56° (horizontal x vertical). The output resolution is up to 1280 x 720 with a frame rate up to 90 fps. Intel has an available tool kit that allows the camera to work on Windows

10, Linux, and Ubuntu 16.04. The Intel RealSense Depth Camera D series has an online database of documentation, which may be applied in determining the configuration and calibration of the camera.



Figure 6. Intel RealSense Depth Camera D435

Advantages	Disadvantages
Optimal range	No heritage
High Resolution and frame rate	Product must be shipped
Many OS options	

Table 8. Advantages and Disadvantages of Intel RealSense D435

4.2.3. Real Sense SR300

The RealSense Camera SR300 was originally created by Intel, and has similar features to the Xbox Kinect. It uses a combination RGB+IR system but with an additional IR laser for depth finding. It was designed specifically for the purpose of 3D imaging and it operates between 0.2 and 1.5 meters for depth. The drawback is that it is much more difficult to communicate with, requiring Windows 10 and a specific Interconnect Cable with SDK software.⁶



Figure 7. RS-SR300⁵

Advantages	Disadvantages
3D camera allows for good depth perception	Harder to acquire
Low cost	Only Windows 10 compatible
Decent resolution and refresh rate	Difficult software integration

Table 9. Advantages and Disadvantages of RealSense SR300

4.2.4. OPAL-P500

The OPAL-P500 was designed by Neptec Technologies, and is the most prominent brand of sensors used by NASA on the International Space Station. It is a powerful and versatile 3D LiDAR sensor that features optimized perception capabilities for detecting small objects at range. Though optimized primarily for long range object detection, the

sensor is still functional at shorter ranges. The OPAL-P500 is user friendly and is fully compatible with 3DRi SDK software. The data measured from the device is a time-stamped position (x,y,z) and respective intensity.



Figure 8. OPAL-P500

Advantages	Disadvantages
Moderate acquisition rate	Difficult to obtain
High accuracy and precision	Limited product information
Wide range of acceptable temperatures	Heavy unit weight

Table 10. Advantages and Disadvantages of OPAL-P500

4.2.5. Snoopy V-Series VUX-IHA

The RIEGL VUX-1HA was designed by RIEGL Laser Measurement Systems. It is a high-performance LiDAR sensor for KINEMATIC laser sensing that offers a 360° field of view with multiple target compatibility. The sensor is, however, only functional after a minimum range of 1.2 meters, limiting its allowable placement with respect to the target object. The RIEGL VUX-1HA is a lightweight scanner with a wide range of acceptable pulse rates with significantly high accuracy and precision.

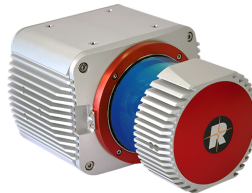


Figure 9. RIEGL VUX-1HA

Advantages	Disadvantages
High acquisition rate	Difficult to obtain
High accuracy and precision	Requires minimum range
Moderate acceptable temperaturerange of range	Light unit wesihline

Table 11. Advantages and Disadvantages of Snoopy V-Series VUX-IHA

4.3. Sensor Data Processing Software

The first major software component of the control loop is the *sensor data processing system*, which must be able to receive data collected by the sensor suite, interpret it, and make two primary determinations. First, it must determine the relative positions and orientations of the grapple point and of the end effector, which must be accurately determined to within 4mm and 5°, as defined in **DR 3.1-3.4**. It must then interface with the control system and provide it with the determined grapple point and end effector states. Furthermore, the sensor processing system must be capable of doing the above with sufficient speed in order for the control system to achieve the desired refresh rate of 2πHz, as given in **DR 1.1**. The capability and accuracy of the system is of paramount importance for two main reasons. First, the grapple point and end effector states it determines are the foundation upon which the control system does its

pathfinding. Second, a precise reading of the end effector's state will allow the entire system to command the arm more accurately: under project KESSLER's implementation, arm position was determined purely through angles reported by the arm's actuators, which introduces a large amount of error. Augmenting the data reported by the actuators with data from the sensor and sensor processing systems will serve to significantly minimize error.

4.3.1. Development

While in the process of developing the algorithms and overarching software package that power the spatial processing system, criteria that should be weighted most heavily include ease of use, suitability in regard to image processing, documentation, and the existence of useful libraries.

MATLAB

At this stage, MATLAB seems like an extremely promising choice. MATLAB is uniquely well-suited to operating on matrices, which is generally the basis for processing images or other types of raw data the sensor suite is likely to provide as input. Furthermore, MATLAB provides supporting libraries like its Image Processing Toolbox, its implementation of the RANSAC algorithm, and its Computer Vision System Toolbox. MATLAB was used for image processing by last year's project KESSLER, proving its potential as a development platform. MATLAB is easy to use, is extremely well-documented, and project members have extensive experience using it. MATLAB has strong visualization capabilities, which may assist in debugging software and ensuring that it's producing accurate results. As elaborated upon below in the discussion on control software, MATLAB features a system (MATLAB Coder) for converting its code to C/C++ code, which may prove useful should either of those languages be chosen for the operational version of the system. As stated on the MathWorks website, much of the Image Processing Toolbox is compatible with MATLAB's C/C++ conversion system. Though typically expensive, MATLAB is available to the team at no cost. Despite these advantages, MATLAB is not without its drawbacks:

- The reliability and capability of the MATLAB Coder conversion system hasn't been tested or quantified; should C or C++ be chosen as the operational programming language and should MATLAB Coder not prove sufficient, unforeseen complexity could be added to the project in the form of having to recode parts of the system in the language chosen.
- If neither C, C++, or MATLAB is chosen as the operational language, the entirety of the system may have to be rewritten in the language chosen.
- Though not as important during the development phase, MATLAB is very slow and has a lot of overhead, owing largely to its status as an interpreted (rather than compiled) language. If translating the developmental MATLAB version of the system into a faster language prove infeasible, cornering the team into using MATLAB in the operational version, it may prove so slow as to excessively hinder the closed-loop control system's operational rate.

Advantages	Disadvantages
Very useful relevant libraries	Unknown difficulty in converting to faster language
Easy to use and debug	Very slow relative to other options
Well-documented, large community	

Table 12. Advantages and Disadvantages of Using MATLAB for the Development of the Sensor Data Processing Subsystem.

C

Another option is to directly write the sensor data processing system in a faster system such as C. C is a lower-level language than MATLAB, is compiled (and is hence much faster), and is generally portable between systems (and can be applied to embedded systems). Developing the sensor processing subsystem in C offers some benefits alluded to above:

- The system would not have to be rewritten for the operational phase, minimizing the risk of encountering unanticipated obstacles and slowdowns as the operational version of the grappling arm comes together.
- The efficiency and speed of the sensor processing system could be reliably tested as soon as the development of the code was completed.

However, team members have less experience with C than with any other option, and C is more difficult to debug than MATLAB and is generally more difficult to work with. A number of image processing libraries are available for C which would likely be useful during the development process, namely the OpenCV and ImageMagick libraries. OpenCV seems like a very promising option as it comes packaged with algorithms designed for things like object recognition and constructing 3D models of objects they identify. C is available at no cost.

Advantages	Disadvantages
No need to convert to faster language	More difficult to debug
Useful libraries available	Team has less experience
	More difficult to use

Table 13. Advantages and Disadvantages of Using C for the Development of the Sensor Data Processing Subsystem.

C++

Another option is to develop the sensor processing subsystem in C++, the extension to C. Though slightly slower than C under some circumstances, it's generally easier to use and comes packaged with a wide variety of useful libraries that C lacks. Several project members are completing or have completed computer science minors and have had relatively significant exposure to C++. If C++ is chosen for development, it will likely be fast enough to stick around for the final version of the arm. C++ is capable of using the same image processing libraries as C. Again, OpenCV seems like an excellent option.

Advantages	Disadvantages
No need to convert to faster language	Slightly slower than C
Easier to develop in than C	More difficult use than MATLAB
Team members have experience with C++	More difficult to debug than MATLAB
Useful libraries available	

Table 14. Advantages and Disadvantages of Using C++ for the Development of the Sensor Data Processing Subsystem.

Python

A final option is to use Python, a dynamic, interpreted language. Python is easier to develop with than C or C++. Project members participating in computer science minors also have experience with Python. Python is available at no cost.

Advantages	Disadvantages
Easy to use	More difficult to debug than MATLAB
Useful available libraries	
Possible to convert to C/C++	

Table 15. Advantages and Disadvantages of Using Python for the Development of the Sensor Data Processing Subsystem.

4.3.2. Operational

Per project KESSLER's spring final review, the majority of the time it took for the system to complete a grappling operation was spent doing visual processing, which took a staggering two minutes. As this year's project will introduce closed-loop feedback to the system, it shall naturally demand that the sensor processing segment of the operation be vastly accelerated. A significant amount of KESSLER's processing time was dedicated to recognizing the point on the satellite their arm was to grapple; by instead assuming a consistently-shaped, external grappling point, it is conjectured that processing time will be significantly reduced.

More importantly, it is presumed that rewriting the visual processing system (previously written in the interpreted MATLAB) into a faster, compiled language will vastly accelerate the data processing time. Thus, during *operation*, considerations such as performance, compatibility, and ease of integration with the sensor suite and control system should be prioritized highest.

MATLAB

Last year's project KESSLER wrote their visual processing system in MATLAB, leading two a single image requiring a full two minutes to process. As this project's most central goal is closed-loop feedback, much faster processing time

is required. MATLAB is an interpreted language with significant overhead and slow execution, which is clearly its primary downside as an operational software platform. However, using it *would* leave its useful toolboxes, namely the Image Processing Toolbox and the Computer Vision System Toolbox, available for use, and would preserve time that would otherwise be lost converting the MATLAB development version of the subsystem to another language. It should also be noted that ROS is written in C++, and thus that writing the operational sensor processing software in MATLAB might create obstacles during integration.

Advantages	Disadvantages
Heritage from KESSLER	Cripplingly slow compared to other options
No need to convert if MATLAB also development language	Inferior compatibility with practical robotics
Useful libraries available	
Easy to debug	

Table 16. Advantages and Disadvantages of Using MATLAB for the Operational Version of the Sensor Data Processing Subsystem.

C/C++

From the wide array of so-called "high-level" (i.e. portable) languages, C and its extension C++ are two of the fastest. No matter which development language is chosen, tools exist that purport to be capable of converting the development version of the software into C/C++. Furthermore, C/C++ are often used in embedded systems, and should SNC continue to develop the project and eventually move it toward a space-worthy system, moving the software in a direction friendly to embedded systems could be of value.

Advantages	Disadvantages
Excellent performance	Difficult to work with
Tools to convert development code to C available	Team members have limited knowledge
Compatible with embedded systems	Difficult to debug

Table 17. Advantages and Disadvantages of Using C/C++ for the Operational Version of the Sensor Data Processing Subsystem.

Python

As an interpreted language, Python is significantly slower than C and C++ and generally isn't as naturally compatible with embedded systems. It's easier develop in than either, but usually just as difficult to debug. As noted earlier, a number of excellent image processing libraries are available in Python, and also as noted, team members have some experience with the language.

Advantages	Disadvantages
Easy to use	Slow, interpreted
Useful libraries available	Relatively difficult to debug

Table 18. Advantages and Disadvantages of Using Python for the Operational Version of the Sensor Data Processing Subsystem.

4.4. Control Software

Once the relative positions of the end effector and grapple point are determined, the closed-loop control system must calculate a path for the end effector to follow. Here path is defined as a series of points in space for the end effector to follow. Once a path is determined, the system must translate the path into commands to transmit to the arm's actuation system. These commands must be generated such that the end effector reaches the grapple point with a maximum error of TBD units. Due to the closed-loop nature of this project, the control software must be able to go from receiving the relative positions as inputs to sending commands to the arm in less than TBD seconds (a direct consequence of the required system refresh rate of TBD Hz). This combined with the complex nature of such a control system leads the team to believe that it will be beneficial to develop the software in a way that makes it easier to debug and quickly iterate the code. Once the control system is proven successful in the development suite of software, the team will translate it to the operational set of software. In the following sections, the main sets of development and operational software choices will be laid out as well as their benefits as relating to the project objectives.

4.4.1. Development

Software: Matlab and Simulink

MATLAB in combination with Simulink provides a graphical programming interface for modeling, simulating, and analyzing dynamical systems. In addition, MATLAB has a large variety of packages that augment the functionality of Simulink to be able to develop control systems as well as simulate and optimize their effects on the modeled system. One of these packages can be used to convert the control system developed in Simulink into C code and other embedded languages. This would greatly simplify the process of converting the developed control system to the operational control system. Finally, Simulink is very widely used and well documented. The combination of Simulink and MATLAB's good documentation, its graphical interface, and the flexibility of tools that can be used with it with it makes Simulink a simple and powerful tool that would help in developing a control system with sufficient accuracy. A con of using Simulink is that MATLAB licences are fairly expensive. However, in this case it is not a problem since the team has already access to student licences of the software.

Advantages	Disadvantages
Easy to use, lots of support libraries	Very slow compared to other languages
Team has plenty of experience	
If Simulink is used, no need to convert	

Table 19. Advantages and Disadvantages of Using MATLAB for Control Software Operations

Software: MSC Systems and Controls

MSC Systems and Controls is another piece of software that is used to simulate and analyze a variety of dynamic systems control systems for them. The Advantages for this choice are that the software includes a graphical interface for modeling the control system, and it seems to have components specialized for robotic arms. The main con of this software is that it specializes in performing simulations and analysis, leaving much to be desired in the realm of motion planning and conversion to operational code. Similarly to Simulink, the software license is not free. However, it is not already owned by the team.

Advantages	Disadvantages
Good user interface	High cost
Good at performing simulations and analysis	Little functionality for converting to operational software
	Team has no experience

Table 20. Advantages and Disadvantages of Using MSC Systems and Controls for Control Software Development

Software: OpenModelica

OpenModelica is an open source modeling and simulation environment based on Modelica. It includes toolboxes for developing models and control systems with a graphical user interface. It also has tools to analyze and optimize the performance of control systems. All of these features make it well suited to developing a control system that has the desired accuracy. Additionally, the software is open source and therefore free. However, the open source nature of this software is also a con since the software is actively receiving updates. This in addition to the relatively small user base means that the documentation is less thorough than the other options. Combined with the fact that none of the team has any experience with Modelica, OpenModelica would have the steepest learning curve out of these options.

Advantages	Disadvantages
No cost	Team has no experience
Plenty of modeling, analysis, and implementation functionality	Lack of documentation
No conversion to operational code required	High learning curve

Table 21. Advantages and Disadvantages of Using OpenModelica for Control Software Development

4.4.2. Operational

Language: MATLAB

Again, MATLAB contends for the operations software language as well, as it provides a ton of official support through MathWorks, MATLAB's developer. Of the many languages available for use, MATLAB is the most user-friendly and the team has by far the most experience with it, offering the benefit of the whole team being able to understand the scripts. From this, it is safe to conclude that this will be the language that will be the simplest to bring to operation. However, though MATLAB may be the easiest to use, it is generally much slower than more low-level programming languages such as C, Python, or Java. As such, it may be necessary to use a more low-level language in order to achieve the closed loop rates laid out in the specific objectives. In terms of compatibility with the robotic operating system (ROS), MATLAB has developed its own software for integrating Simulink with ROS. And with all of this, MATLAB makes a suitable language for operations in this project.

Advantages	Disadvantages
Easy to use, lots of support libraries	Very slow compared to other languages
Team has plenty of experience	
If Simulink is used, no need to convert	

Table 22. Advantages and Disadvantages of Using MATLAB for Control Software Operations

Language: C

C is a lower level programming language than any of the alternatives that can be optimized to the architecture of the chip it is being run on. Its major benefit is that in general it has the fastest computation time out of the options laid out here. As a result of C being a lower level language, it is the most difficult to debug out of the choices laid out here. There is not much official support in place to help the user move past pitfalls and confusion. There is, however, much support available through external open-source resources on websites like GitHub or tutorials like those offered for free by Oxford University. Also noteworthy, C was the primary operational language used in the KESSLER project from a previous year whose code repository will provide a useful reference.

Advantages	Disadvantages
The fastest language in consideration	High learning curve, hard to debug
Many open-source support libraries	Long, complicated code scripts
Versatile, can optimize for different needs	Fewest official support libraries
Heritage from previous years	

Table 23. Advantages and Disadvantages of Using C for Control Software Operations

Language: C++

C++ is an iteration of the language C discussed above; this iteration was designed to be much more friendly to users than C while still maintaining the lightweight, compiled nature of the C language. Because of this, its performance in embedded systems like robotics is hindered slightly by added overhead while its application to large processing systems is buffed.³ Also, with the added efficiency of supporting pointers and more official support libraries, C++ brings more infrastructure to the table than C. So with each pro of C++ comes a con in comparison to C in reference to our robotics control system. Last, there is not as much heritage with C++ for reference as with C. It is important to note that C and C++ are so closely compared to each other in this section because C++ is so derivative of C that the two can be compared to the other languages as a unit and then chosen between if necessary.

Advantages	Disadvantages
Added infrastructure, official libraries	Added infrastructure marginally slows C++ compared to C
More user-friendly than C	Not as tailored for simple robotic systems as C
Maintains low-level, lightweight, compiled status	

Table 24. Advantages and Disadvantages of Using C++ for Control Software Operations

Language: Python

More popular in the wide world of programming is Python, a more medium-level programming language that has become more widely used in recent years than most other languages.⁴ With this widespread use and popularity comes plenty of open-source and official support for all kinds of systems, including embedded robotics systems like this one; Theano is one example. These libraries are designed to tailor Python's efficiency to each specific use, bringing its speed up, though its speed still does not match that of C/C++. And, previous iterations of this project have used Python for pieces of their software package, giving our team a possible reference. The major benefit of using Python would be its wide usage and support availability that matches MATLAB and far outshines C/C++ in terms of ease of use, error resolution, and readability.

Advantages	Disadvantages
Easy to read and write, lots of support	Slower than C/C++, faster than MATLAB
Support libraries tailored to all sorts of uses and needs	
Some heritage from previous years for reference	

Table 25. Advantages and Disadvantages of Using Python for Control Software Operations

4.5. Hardware

4.5.1. Grappling Point Manufacturing

With one key component of this project being the design and manufacture of a specific grappling point comes the problem of what material to use and how to make it. A separate trade study will discuss the best possible design of the grappling point, so only the material and manufacturing process will be discussed in this subsection. After much research, four total options were selected for comparison: 3D printing, Lego's, Cardboard, and purchase of a pre-made grappling point.

3D Printed Object

The 3D printer which would be used to manufacture the grappling point is the Lulzbot Taz 6. This printer is readily available on campus and is extremely low cost. PolyliteTM PLA is the specific filament used on these 3D printers which is a Poly(lactic acid) resin. This material comes in a wide variety of colors and has a density of 1.25 g/cm³. PLA has a melting point of 140°C, and the Taz 6 has a maximum print bed area of 11 x 11 inches. The TAZ 6 prints using Fused Deposition Modeling (FDM) to create parts, which is the process of laying material down in layers which then fuse together creating one solid object.

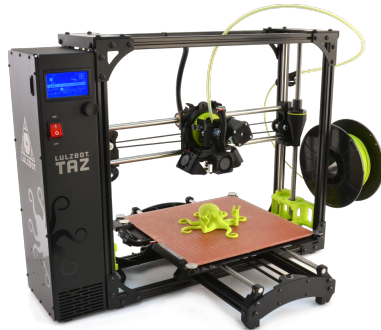


Figure 10. Lulzbot TAZ 6



Figure 11. Polylite PLA

Advantages	Disadvantages
Cheap (potentially free)	Time to manufacture
Wide range of colors	PLA can break if dropped
Can print any shape or design	

Table 26. Advantages and Disadvantages of 3D printing.

Lego's

Lego's are plastic toy blocks which come in a variety of shapes and colors. They can quickly and easily be rebuilt if broken, which is a major reason they were chosen as a manufacturing option, since the grappling maneuver poses strong risk of running into, crushing, or dropping the grappling point.

Advantages	Disadvantages
Quick manufacture	Easily broken if dropped or crushed
Wide range of colors	Difficult to construct certain designs
Available in multiple sizes	Cost associated with specific pieces

Table 27. Advantages and Disadvantages of using Lego's.

Cardboard

Cardboard is an extremely common material used in everyday life from toilet paper rolls, to storage boxes and more. There are different types of cardboard including, corrugated cardboard, flat cardboard, paperboard and many more. Cardboard is cheap and easily attainable from old packaging or found in recycling bins. It is also very lightweight but can easily be dented or crushed. Another issue may be it's ability to form different shapes due to its stiffness.

Advantages	Disadvantages
Cheap (potentially free)	Weak, easily crushed under too much force
Lightweight	Only one color but could be painted
Ability to recreate multiple objects	Difficult to manufacture certain designs
Recyclable	

Table 28. Advantages and Disadvantages of Cardboard.

Purchase of Pre-Made Grapple Point

The last manufacturing option technically involves no manufacturing at all. Once a grapple point design is selected a pre-made object could be purchased. For example if a cube or cylinder were chosen at the end of the design trade study then children's toy blocks could be purchased in the desired shape and color. The problem with purchasing a grapple point is that it can quickly become very costly and the exact design dimensions may not be available.



Figure 12. Pre-made possible grapple points

Advantages	Disadvantages
No manufacturing required	Costly
Quick and easy replacement	Limited sizes and shapes
	Personalized designed unavailable

Table 29. Advantages and Disadvantages of purchasing a pre-made grapple point.

4.5.2. Grapple Point/Object Design

Determining the grapple point on the satellite for the end effector to grapple is a vital factor for the rest of the mission. The grapple point drives the rest of the process for successfully acquiring the satellite. The sensors and coding programs being used for determination of the grapple point need a base to search for to continuously analyze the optimal path and arm actuation. While most of this process is done with the software and visual systems, the grapple point is important so the sensors can easily identify the grapple point in order to carry out the mission. Choosing the design for the grapple point will determine the functionality of the software used to visually identify the grapple point, therefore the following criteria was designed to help determine which grapple point design is optimal for the application of the mission.

Flat Plate

Designing the grapple point to be a flat plate would be a relatively simple design. The flat plate is designed to be a thin structure. The flat plate will contain simple framework and, depending on material chosen, is optimal for compressible loading. The efficiency of the flat plate is also optimal as it has a small volume but a large surface area. A flat plate has its advantages and disadvantages seen below in table 30. The Advantages and Disadvantages developed in this table will be used to help determine the weight of the design in the trade studies conducted in the following sections of the report.

Advantages	Disadvantages
Easy to manufacture	Difficult for sensing
Could be used as solar array	Specific grapple orientation
Great surface area	

Table 30. Advantages and Disadvantages of using a flat plate as the grappling point.

Cylinder

Designing the grapple point to be a cylinder is an obscure design for the satellite. The design of the cylinder would be designed to be representative of a door handle. This will enable a few end effector designs to get a closed grip around the grapple point. However, the design also limits the application of force sensors determining if the end-effector has accomplished a successful grapple. Advantages and disadvantages for a cylinder grapple point are noted in table 31 and will be used to help determine the weight of the design for the trade studies in the following sections.

Advantages	Disadvantages
Allow for safer and more successful grapple	Limited use of pressure and load sensors
Simple design for manufacturing	Less surface area

Table 31. Advantages and Disadvantages of Using a Cylinder as the Grappling Point.

Knob

Using a knob as a grapple point is not optimal for any end-effector design. The knob design is focused towards the appendages end-effector designs. The knob will be difficult to manufacture and have a small surface area, making it difficult for the pressure sensors to detect loading. Like the flat plate design, the knob is designed to have a high compressible load, as well as a moderate tensile load capability. Advantages and Disadvantages are shown below in table 32 and will be used to help determine the weight of the design for the trade studies in the following sections.

Advantages	Disadvantages
Easily identifiable by sensors	Only optimal for multiple appendages end-effector
	Difficult design for manufacturing

Table 32. Advantages and Disadvantages of Using a Knob as the Grappling Point.

Ring

Using the ring as a grapple point really limits the design of the end-effector. While the ring provides an easy solution for the hook end-effector, it limits the use of other end-effector designs. The ring will be designed to be a uniform circle with connections at the end to attach to the host vehicle. The ring allows for high tensile strength as it would be pulled by a hook, focusing its load on one point of the ring. Advantages and Disadvantages are depicted below in table 33 and will be used to help determine the weight of the design for the trade studies in the following sections.

Advantages	Disadvantages
Easy to manufacture	Limited to hook end-effector
High tensile load capabilities	
Easily identifiable by sensors	

Table 33. Advantages and Disadvantages of Using a Ring as the Grappling Point.

Cube



Similar to the flat plate design, the cube would also be a simple design. The cube will be relatively small and fit easily in a majority of end-effector's grips. Having a cube grapple point also allows for symmetrical loads as it is symmetrical on all sides. Like the flat plate, the cube will have a greater surface area and can withstand a greater compressible load factor. The issue with the cube as a grapple point is that the cube will allow for less margin error as it has a greater volume than the other grapple points. Advantages and Disadvantages are detailed below in table 34 and will be used to help determine the weight of the design for the trade studies in the following sections.



Advantages	Disadvantages
Able to be grappled by most end-effector's	Low margin of error
Symmetrical loading capability	
Great surface area	


Table 34. Advantages and Disadvantages of Using a Cube as the Grappling Point.


4.5.3. Arm Design

The CrustCrawler arm has many different girders, servos, turntables, and wrist joints to customize an arm design. The girders vary in length. The servos vary in size, weight, and stall torque. The turntables vary in stall torque. The wrist joints vary in size and stall torque. The specs for these parts are shown in figure 13.

	MX/RX-28 Single Axis Kit Height = 2.91in.(7.39cm) Width = 1.89in.(4.8cm) Weight = 3.4 oz.	(4) #4 Lock nuts (4) #4- 1/4in. Screws (4) M2.5 – 8mm screws/ nuts (4) #4 .31in countersink screws
	MX/RX-64 Single Axis Kit Height = 3.50in.(8.89cm) Width = 2.21in.(5.6cm) Weight = 6 oz.	(4) #4 Lock nuts (4) #4- 1/4in. Screws (4) M2.5 – 8mm screws / nuts (4) #4 .31in countersink screws
	MX-106 Single Axis Kit Height = 3.6in (9.14cm) Width = 2.45in. (6.22cm) Weight = 6.9oz.	(4) #4 Lock nuts (4) #4- 1/4in. Screws (4) M2.5 – 8mm screws/ nuts (4) #4 .31in countersink screws
	MX/RX-28 Dual Axis Kit Height = 3.10in.(7.87cm) Width = 4.94in.(12.54cm) Weight = 8.10oz.	(16) #4- .31" countersink screws (4) #4-1/4" screws (8) #4- Lock Nuts (8) M2.5 – 8mm screws / nuts
	MX/RX-64 Dual Axis Kit Height = 3.68in.(9.34cm) Width = 5.10in.(12.95cm) Weight = 13.2oz.	(16) #4- .31" countersink screws (4) #4-1/4" screws (8) #4- Lock Nuts (8) M2.5 – 8mm screws/ nuts

	DESCRIPTION	HARDWARE
	2.5in (6.35cm) Girder Weight - .8oz.	(12) #4 Lock nuts (4) #4 - 1/4" screws (8) #4- .31in. Countersink Screws
	5 in. (12.7cm) Girder Weight - 1.3oz.	(12) #4 Lock nuts (4) #4 - 1/4" screws (8) #4- .31in. Countersink Screws
	Adjustable 8.31in. (21.11cm) to 13.21 in. (33.81cm) Girder Weight - 3.9oz.	(12) #4 Lock nuts (4) #4 - 1/4" screws (8) #4 - .31in. Countersink Screws
	Single Axis Adapter Plate This plate is required any time a single axis is going to be connected to one end of the girder. Weight - .6oz.	(4) #4 - .31in. Countersink Screws

PART #	DESCRIPTION	HARDWARE
	MX/RX-28 Turntable Dimensions - 4.5" X 4.5" (11.43cm X 11.43cm) Mounting Tabs - 5.25" (13.33cm) center to center	(4) M2 8mm countersink screws (4) M2 10mm screws (1) Turntable spacer (1) Turntable bearings set
	MX/RX-64 Turntable Dimensions - 4.5" X 4.5" (11.43cm X 11.43cm) Mounting Tabs - 5.25" (13.33cm) center to center	(4) M2.5 8mm countersink screws (4) M2.5 10mm screws (1) Turntable spacer (1) Turntable bearing set
	MX-106 Turntable Dimensions - 4.5" X 4.5" (11.43cm X 11.43cm) Mounting Tabs - 5.25" (13.33cm) center to center	(4) M2.5 8mm countersink screws (4) M2.5 - 10mm screws (1) Turntable spacer (1) Turntable bearing set

	AX12A / AX-18A Wrist Rotate Weight = 3.5oz.	(4) M2 – 14mm screws (12) #4 – 1/2" screws (4) #4 lock nuts (2) AX Side brackets (2) RX/MX-28 angle brackets (1) .394" (9.93mm) Axis Spacer
	MX-28 Wrist Rotate Weight = 4.5oz.	(4) M2 – 14mm screws (8) M2.5 – 8mm screws (12) #4 – 1/2" screws (4) #4 lock nuts (2) MX Side brackets (2) RX/MX-28 angle brackets (1) .394" (9.93mm) Axis Spacer
	MX-64 Wrist Rotate Weight = 6.6oz.	(4) M2 – 14mm screws (8) M2.5 – 8mm screws (12) #4 – 1/2" screws (4) #4 lock nuts (2) MX Side brackets (2) RX/MX-64 angle brackets (1) .394" (9.93mm) Axis Spacer


	MX-106 Dual Axis Kit Height = 3.79in. (9.62cm) Width = 5.32in (13.51cm) Weight = 15oz.	(16) #4- .31" countersink screws (4) #4-1/4" screws (8) #4- Lock Nuts (4) M2.5 – 8mm screws/nuts
---	--	---

Figure 13. Specs for CrustCrawler components.

Using these components three different arm designs were created. Properties considered while designing were range of motion (degrees of freedom), length, weight, servo stall torque, and complexity. While we want a large range of motion, too many joints can result in more complexity for the closed loop software problem.

4.5.4. Design 1

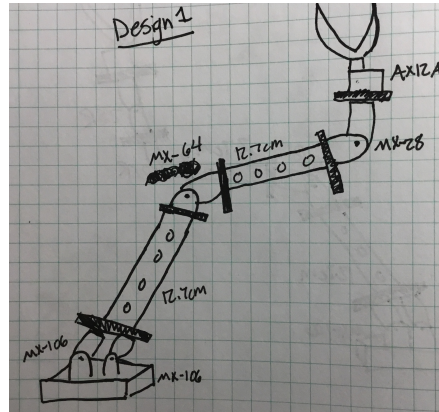


Figure 14. Arm Design 1

The first arm design is shown in figure 14. The first design is described from base to claw as:

- MX-106 Turntable - \$650
- MX-106 Dual Axis servo - \$1183
- 12.7cm girder - \$34
- MX-64 servo kit - \$370
- 12.7cm girder - \$34
- MX-28 servo kit - \$280
- AX12A Wrist joint - \$109
- End effector
- Total Cost - \$2,660

This arm was designed with simplicity in mind. The arm has five joints giving it a fairly wide range of motion while having less complexity for the software to consider. Because the arm has less servos, it weighs less. The lightweight will keep the risk of breaking or overheating servos low. The lack of servos also drives the cost of the arm down.

Advantages	Disadvantages
Simple for software	Less range of motion
Low risk of servo failure	
Low cost	

Table 35. Advantages and Disadvantages of Arm Design 1

4.5.5. Design 2

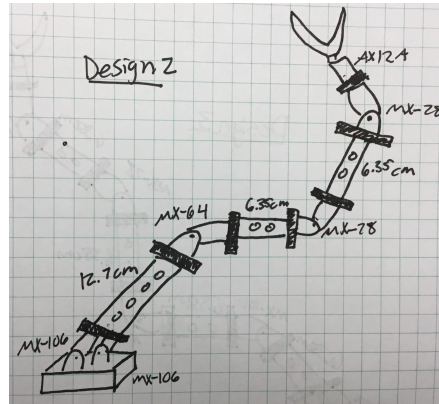


Figure 15. Arm Design 2

The second arm design is shown in figure 15. This design is described from base to claw as:

- MX-106 Turntable - \$650
- MX-106 Dual Axis servo - \$1183
- 12.7cm girder - \$34
- MX-64 servo kit - \$370
- 6.35cm girder - \$28
- MX-28 servo kit - \$280
- 6.35cm girder - \$28
- MX-28 servo kit - \$280
- AX12A Wrist joint - \$109
- End effector
- Total Cost - \$2,962

This arm was designed to have slightly larger range of motion than Design 1. The arm has six joints giving it a wide range of motion while having adding more complexity to the software problem. The arm does weigh slightly more, but the servos are still unlikely to fail or overheat. The cost of adding another servo is considerable at around \$300.

Advantages	Disadvantages
More range of motion	More complex for software
Low risk for servo failure	More expensive

Table 36. Advantages and Disadvantages of Arm Design 1

4.5.6. Design 3

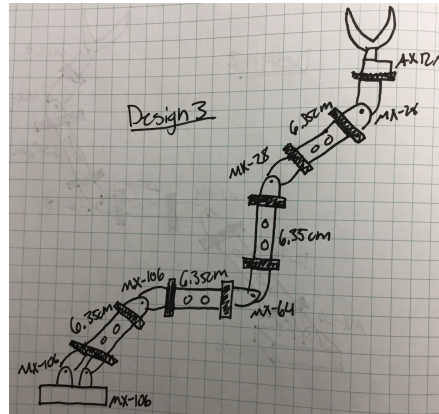


Figure 16. Arm Design 3

The third arm design is shown in figure 16. This design is described from base to claw as:

- MX-106 Turntable - \$650
- MX-106 Dual Axis servo - \$1183
- 6.35cm girder - \$28
- MX-106 servo kit - \$563
- 6.35cm girder - \$28
- MX-64 servo kit - \$370
- 6.35cm girder - \$28
- MX-28 servo kit - \$280
- 6.35cm girder - \$28
- MX-28 servo kit - \$280
- AX12A Wrist joint - \$109
- End effector
- Total Cost - \$3,547

This arm was designed to have a very large range of motion. The arm has seven joints giving it a wide range of motion while adding much more complexity to the software problem. The arm does weigh significantly more than Design 1. As seen above, a few of the servos were upgraded to try to keep the risk low. However, the upgraded servos do add significant weight and cost.

Advantages	Disadvantages
Very wide range of motion	More complex for software
	More expensive
	Higher risk for servo failure

Table 37. Advantages and Disadvantages of Arm Design 3

4.5.7. End Effector

The end effector is the device which will be used at the end of the MEGACLAW arm to interface with the target grapple point and accomplish the target grapple. As key success criteria will depend on the establishment of a successful grapple, the end effector which is chosen is of utmost importance. Below the four end effector design options considered are briefly described.

Claw (Clamp)



Figure 17. Crust Crawler "Dual Gripper" Clamp Style End Effector¹⁶

The claw (clamp) end effector design grapples a target by applying a force through each appendage to grasp and stabilize a target grapple point. Each appendage may move in one degree of freedom through flexion of appendage joints in order to interface with various grapple points. In the case of a flat plate grapple point force would be applied to each side of the plate. Additionally, appendages may be used to enclose a cylindrical or ring type grapple point. This design was used in the heritage predecessor project KESSLER.

Advantages	Disadvantages
Compatible with the Flat Plate, Cylinder and Ring grapple point designs	Utilizing appendages with actuated joints introduces the opportunity for joint failure
Heritage projects allow for immediate system integration	
High grip strength and target stability once grappled	

Table 38. Advantages and Disadvantages of Claw (Clamp) End Effector

Multiple Appendages

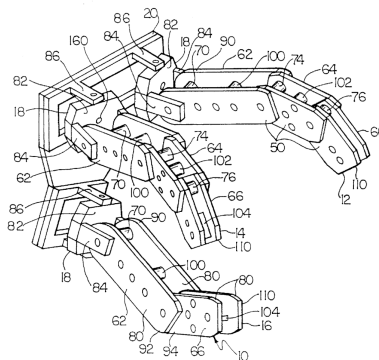


Figure 18. Multiple Appendages Style End Effector¹⁷

A multiple appendages end effector approach works much like a claw (clamp) end effector with an increased number of appendages to increase adaptability to grapple points. Each appendage may move in two degrees of freedom (rotation at the appendage base and flexion of appendage joints) in order to interface with various grapple points. Force is applied through each appendage to grasp and stabilize a target grapple point in the case of a flat plate or knob grapple point. Additionally, appendages may be used to enclose a cylindrical or ring type grapple point.

Advantages	Disadvantages
Compatible with the Flat Plate, Cylinder, Knob and Ring grapple point designs	Increasing the number of appendages and corresponding joints and hinges increases the opportunity for joint or hinge failure
	Limited commercial off-the-shelf options
	High cost

Table 39. Advantages and Disadvantages of Multiple Appendages End Effector

Interlacing Appendages

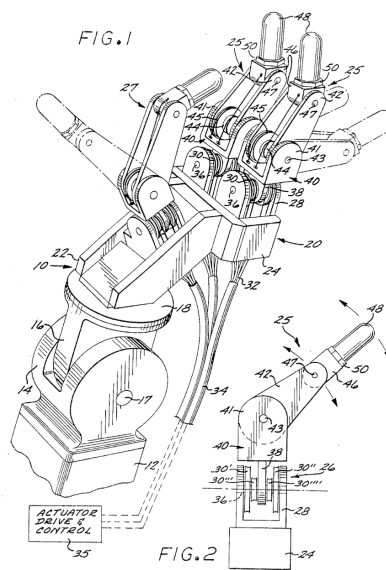


Figure 19. Interlacing Appendages Style End Effector¹⁸

An interlacing appendages end effector approach works to grapple a target by encapsulating the target within the appendages of the end effector. Each appendage may move in one degree of freedom through flexion of appendage joints in order to encapsulate a grapple point. Appendages would be used to enclose a cylindrical grapple point. Additionally, it would be possible to enclose a ring type grapple point.

Advantages	Disadvantages
Compatible with the Cylinder and Ring grapple point designs	Increasing the number of appendages and corresponding joints and hinges increases the opportunity for joint or hinge failure
Ideal end effector for a cylinder shaped grapple point	Limited commercial off-the-shelf options
Target will not disengage from end effector once encapsulated barring a joint failure or grapple point failure	High cost

Table 40. Advantages and Disadvantages of Interlacing Appendages End Effector



Figure 20. Closing Hook Style End Effector

The closing hook end effector design grapples a target by engaging a spring arm and then containing the target grapple point within the encapsulating hook. The target grapple point pushes against the spring arm (which ideally would be operated with an actuator), then sliding into the encapsulating hook at which point the spring arm moves back into the original position diagrammed below.

Advantages	Disadvantages
Low complexity - No actuators or motor required. Little to no chance of failure	Force required to engage the spring arm would almost certainly alter the orientation of the target grapple point
Ideal end effector for a Ring shaped grapple point	Low stability of target after grapple achieved
Target will not disengage from end effector once encapsulated barring a grapple point failure	

Table 41. Advantages and Disadvantages of Closing Hook End Effector

5. Trade Study Process and Results

5.1. Sensor Trade Studies

5.1.1. Sensor Location

The choice of the sensor location has an effect on the optics, the controls, and the manufacturing of the project and it therefore is worthy of a trade study. The sensor locations were chosen because they were the three forerunners in the discussion on camera location within the team. They were ranked by scoring them according to the following criteria.

- **Visibility**
 - This parameter represents both the field of view provided by the location and takes into account how much of its view will be blocked by the arm. A perfect score in this category would be able to view both the grapple point and the end effector at all times throughout the grapple sequence.
 - This is weighted as 40% because having constant vision of the grapple point and end effector is paramount when attempting to form a closed loop feedback system involving a sensor. If there is not a constant view of either of these objects then no feedback can be given and the arm will have to move blindly until vision is reestablished.
- **Interference (Risk to Arm & Sensor)**
 - This parameter is a measurement of how little risk and modification this position requires. It takes into account if there will be a risk of wires twisting or the arm motion being limited due to the sensor being in the way. This also accounts for the risk of damage to the sensor for each position. A perfect score would require the sensor position to not interfere with the arms motion in any way and have no risk of collision with the end effector and grapple object.
 - This is weighted as 40% because having to design a controls system around a sensor significantly increases the difficulty of this project, and the risk of destroying the camera during testing also must be considered as it could delay the project significantly.
- **Manufacture**
 - This parameter represents both how difficult it will be to install the sensor in this location and what modifica-

tions to the arm will be required for the arm to continue to function. This would include the need for additional actuators. A perfect score would require only trivial manufacturing and would require no modifications to the arm.

-This is weighted as 20% because while it does complicate the other portions of this project, the amount of work created by a harder position is worth having a constant feedback setup as having a constant feed back loop is the objective of this project.

	0-2	3-6	7-9	10
FOV	Completely Obscured	Mostly Obscured	Partly Obscured	Unobscured
Interference	Complete Interference	Some Interference	Little Interference	No Interference
Manufacture	Very Difficult	Some Difficulty	Little Difficulty	Minor Difficulty

Table 42. Scoring Parameter Scale for Sensor Locations

This is where we talk about each option and why we gave what scores we gave.

- **FOV**

- At End Effector: The FOV at the end effector allows for a fuller and closer view of the target object. This resulted in receiving a score of 9.
- On Arm: The FOV of a sensor placed on the arm, faces the challenge of potentially having its view blocked by the arm itself. Because of this, a score of only 6 was assigned to it.
- At Base: A sensor located at the base of the arm would have a clearer view of the target, however might still face challenges with the arm blocking its view. This cause it to receive a slightly higher score than on the arm, by receiving a 7.
- Offset: A sensor located offset from the base of the arm would have a clear field of the target object since it is unlikely for the arm to be able to block its view. This resulted in a score of 9.
- At End Effector and Offset: A sensor located on the end effector and offset from the base of the arm would offer the best field of view of the object since it allows for two perspectives to accurately track the object simultaneously. Because of this combination, this positioning was given the highest score, a 10.

- **Interference**

- At End Effector: A sensor placed on the end effector would face the challenge of possibly being damaged during an arm actuation from grappling the target. This would ultimately limit the configuration and coding of the arm actuation, to account for the added sensor at the end of the arm. This resulted in receiving a low score of 2.
- On Arm: A sensor place on the arm would face similar challenges, however since it is further from the grappling mechanism, it would face less risk to damaging the sensor during arm actuation. This consideration resulted in receiving a score of 4.
- At Base: A sensor located at the base of the arm, would cause no interference to the actuation of the arm. This resulted in receiving the highest score, a 10.
- Offset: As stated above, a sensor located at offset from the base of the arm, would cause no interference to the actuation of the arm. This also resulted in receiving the highest score, a 10.
- At End Effector and Offset: A sensor located at the end effector and offset from the base, would inherit the interference problems that came with just having a sensor on the end effector. Therefore it received the same score, a 2.

- **Manufacture**

- At End Effector: Installing a sensor on the end effector would pose the problems of affecting the weight limitations of the arm, and would cause difficulty of wiring the camera up the length of the arm. Because of this, it resulted in receiving a score of 2.

- (b) On Arm: As stated above, installing a sensor on the arm would pose the problem of potentially affecting the arms weight limitations, and may be difficult to wire up part of the length of the arm. Because an installation on the arm poses similar, but less significant concerns as mounting a sensor on the end effector, it received a slightly higher score of 3.
- (c) At Base: Installing a mount for the sensor at the base would be fairly feasible, especially since the heritage from prior years, used this same location. Because of this, this location received a 9.
- (d) Offset: Installing a sensor offset from the base of the arm would be fairly feasible as well since the mounting wouldn't have to integrate with the base of the arm during testing.
- (e) At End Effector and Offset: Installing a sensor at the end effector and offset from the base would be the most difficult sensor positioning to manufacture. This would inherit the installation consideration for just the end effector above, as well as requiring installation of a sensor offset from the body. Therefore it received the low score of 1.

	Weight	At End Effector	On Arm	At Base	Offset	At End + Offset
FOV	40%	9	6	7	9	10
Interference	40%	2	4	10	10	2
Manufacture	20%	2	3	9	9	1
Weighted Total	100%	4.8	4.6	8.6	9.4	5.0

Table 43. Trade Study for Sensor Location.

5.1.2. Sensor Type

These candidates were then chosen based on their ranking in the following criteria, and they were then charted in the table below.

- **Optimal Distance**
This is with regards to how well the sensor operates from a range of 0.5-2.0 meters. A high score would be assigned to a unit that is optimized for this range specifically.
- **Data Quality**
This parameter represents the degree to which an object can be identified by each sensor. The higher the resolution and refresh rate, the better the score. Additionally, how easy it is to determine the distance an object is away from the sensor.
- **Documentation**
The documentation of other projects attempting similar tasks which gives an idea of how successful other projects have been using each sensor type. The heritage of this project in particular, which analyzed the use of two RGB+IR cameras and a pure RGB camera.
- **Interfacing**
This compatibility of a sensor to connect with various, user-friendly software. A high score would be assigned to a unit if it is highly versatile and compatible with a large array of software packages.
- **Lighting Requirements**
This parameter represents how specific the lighting needs to be for the sensor to operate. A perfect score would be a sensor which can operate at any light level without any adverse effect, and the worst score would be a sensor which only functions in extremely specific lighting conditions.
- **Cost**
Cost is the listed price of the sensor. The higher the score, the cheaper the price of the sensor is. This requires taking into consideration the \$ 5,000 allocated budget.

	0-2	3-6	7-9	10
Optimal Range	Poorest	Sub-optimal	Nearly Optimal	Optimal
Data Quality	Poor Quality	Adequate Quality	Significant Quality	Highest Quality
Documentation	None	Adequate	Significant	Full Documentation
Interfacing	Incompatible	Some Compatibility	Mostly Compatible	Fully Compatible
Lighting Requirements	High Requirement	Moderate Requirement	Low Requirement	No Requirement
Cost	Prohibitive	Expensive	Cheap	Free

Table 44. Scoring Parameter Scale for Sensor Type

- Optimal Range
 - (a) Microsoft Kinect V2: The range of the Kinect is between 0.5 and 4.5 meters which fits the described optimal range. This results in a score of 10.
 - (b) Intel RealSense D435: The range of the Intel D435 0.2 and 10 meters which fits the described optimal range. This results in a score of 10.
 - (c) RealSense SR300: The given operational range is from 0.5 to 2.0 meters, which could use to work for a slightly shorter range but is almost perfect for our application.
 - (d) OPAL-P500: The optimal distance for the OPAL-P500 is not specified, however most LiDAR detectors only operate accurately after 1 m from the sensor. Because of this, it receives a score of 5.
 - (e) Snoopy V-Series VUX-1HA: The optimal distance for the VUX-1HA is listed as requiring a minimum range of 1.2 m from the object to the sensor. Because of this, it receives a score of 4.
- Data Quality
 - (a) Microsoft Kinect V2: The depth resolution is 512 x 424 with a frame rate of 30 fps. This resolution was enough for the KESSLER team to succeed. This results in score of 7.
 - (b) Intel RealSense D435: The output resolution is up to 1280 x 720 with a frame rate up to 90 fps. This resolution and frame rate are much higher than the Microsoft Kinect. This results in a score of 8
 - (c) RealSense SR300: The data quality of this sensor is comparable to the Kinect V2, but it has advantages in ways that are not applicable to this project and is slightly disadvantaged in ways that are important. The refresh rate is slightly higher but the pixel density is worse.
 - (d) OPAL-P500: The OPAL-P500 has a high acquisition rate of 300,000 points/sec and has an accuracy and precision that is less than 20 mm when measured at a distance of 12 m. This results in receiving a score of 9.
 - (e) Snoopy V-Series VUX-1HA: The VUX-1HA has an even higher acquisition rate of 1,000,000 points/sec and has an accuracy less than 5 mm and a precision less than 3mm when measured at a distance of 30 m. Because of this high accuracy and precision, it receives a 10.
- Interfacing
 - (a) Microsoft Kinect V2: The available developer tool kit allows easier compatibility with computer operating systems than sensors that do not have available tool kits. The Microsoft Kinect is only compatible with Windows 10. This results in a score of 8.
 - (b) Intel RealSense D435: The available developer tool kit allows easier compatibility with computer operating systems than sensors that do not have available tool kits. The Intel D435 is only compatible with Windows 10, Linux, and Ubuntu 16.04. This results in a score of 9.
 - (c) RealSense SR300: This sensor interfaces using a standard USB 2.0 but it only works with Windows 10. It is also compatible with the software options of the software team.
 - (d) OPAL-P500: The company was very unspecific in their documentation so it is assumed to be poor.
 - (e) Snoopy V-Series VUX-1HA: The company was very unspecific in their documentation so it is assumed to be poor.

- Lighting Requirements

- (a) Microsoft Kinect V2: The Microsoft Kinect requires a well lit room to operate accurately. The Kinect is able to combine the the RGB sensor with the IR sensor to produce an image with no light. This results in a score of 5.
- (b) Intel RealSense D435: The Intel D435 requires a well lit room to operate accurately. This results in a score of 4.
- (c) RealSense SR300: This sensor requires fairly specific lighting conditions as the IR camera is not as powerful as some other options and it is much worse than the LiDAR.
- (d) OPAL-P500: The OPAL-P500 uses LiDAR which means that it does not require a specific lighting environment. Reflection could however be a concern if in direct sunlight, however, the mission is designed to operate in eclipse around the earth. Because of this, it receives a 10.
- (e) Snoopy V-Series VUX-1HA: Like the OPAL-P500, the VUX-1HA uses LiDAR which also does not require a specific lighting environment. Reflection could however would again, be a concern if in direct sunlight. Because of this, it receives a 10.

- Cost

- (a) Microsoft Kinect V2: The Kinect is provided by Sierra Nevada Corporation. The free sensor results in a score of 10.
- (b) Intel RealSense D435: The Intel D435 cost \$180. This is a reason price for a project of this size. Comparing to a Kinect (free), the D435 is less cheap. This results in a score of 8.
- (c) RealSense SR300: The SR300 costs \$150 which is a reasonable price. The sensor is not clearly available resulting in a score of 5.
- (d) OPAL-P500: The company was very unspecific in their documentation so it is assumed to be very expensive.
- (e) Snoopy V-Series VUX-1HA: The company was very unspecific in their documentation, so it is assumed to be very expensive.

	Weight	Kinect V2	RealSense Depth-D435	RealSense SR300	OPAL-P500	Snoopy V-Series VUX-1HA
Optimal Distance	20%	10	10	9	5	4
Data Quality	30%	7	8	6	9	10
Documentation	15%	9	8	5	3	5
Interfacing	20%	8	9	6	6	6
Lighting Requirements	10%	5	4	5	10	10
Cost	5%	10	8	5	2	2
Weighted Total	100%	8.05	8.2	6.30	6.45	6.85

Table 45. Trade Study for Sensor Type.

5.2. Software Trade Studies

5.2.1. Sensor Data Processing Software

When quantifying the value of each programming language considered for the sensor processing subsystem, the numeric quantities presented in presented in table 46. For each criteria, weights were chosen as functions of each criterion's relative importance:

- Prior knowledge: The knowledge and experience project members have with each option is an importance consideration. The more exposure to a language, the less time the total coding process will take. However, the languages considered all have similar structures and learning more about how to use any one of them isn't anticipated to be especially hard or time intensive. Thus, it's given a weight of 0.15.

- **Ease of Use** - The score for this parameter is based on how well documented the software is, how easy it is to debug, and how much manufacturer or community support there is for the option. Being strong in all of these factors will make the development of the control system much easier, hopefully leading to a more refined result. For this reason, ease of use is weighted heavily, with a value of 0.25.
- **Suitability** - Different programming languages and platforms are more or less suitable to the specific problem of processing the data provided by the sensor system. The suitability of a choice depends on its syntax, structure, and the availability of libraries that could be applied to processing sensor data. Suitability is an important criteria, as choosing an unsuitable choice would mean spending time fighting the software to make it accomplish the desired tasks, or rewriting libraries that might already be available in a more suitable choice. A language's suitability to a specialized task is critically important, and thus it's given the highest weight of 0.3.
- **Convertibility** - Since the operational code will likely be in a different language than the development software, it will be of paramount importance that the team is able to easily and accurately convert from the developed system to the operational language. For this reason convertibility has the largest weight, at 0.2.
- **Cost** - The more something costs for the project, the lower its score. The team's budget is fairly open, so this does not have very high weight, with a value of 0.1.

Note that a language's speed or performance isn't considered in table 46. While developing the pieces of software that will ultimately form the complete system, the performance of any one of those pieces isn't a concern. Only when the entire, functioning grappling arm system begins coming together will performance be a significant consideration, by which point the sensor data processing system (and the control system, discussed below) will likely have been converted into more efficient languages.

	0-2	3-6	7-9	10
Prior Knowledge	None to minimal	Introductory to basic	Sufficient to extensive	Expert
Ease of Use	No support, difficult to read/write	Open-source support	Official support, relatively straightforward to read/write	Well-documented, official support, easy to read/write
Suitability	Interpreted	Compiled, high-level	Compiled, lightweight	Compiled, basic
Convertibility	Non-convertible to highly inconvenient (to transfer to operational language[s])	Relatively difficult to convert, few tools available	Straightforward to convert, many tools available	Conversion inconsequentially easy, or unnecessary
Cost	Maximal (> \$2500)	\$500 – 2500	\$1 – 499	Free

Table 46. Scoring Parameter Scale for Sensor Data Processing Software Choices (Development)

For each criteria presented in table 46, the options for sensor data processing were scored in the following ways.

- **Prior Knowledge**
 - (a) **MATLAB**: The entire team has used MATLAB for almost every lab over the course of their undergraduate studies. The team has no experience using MATLAB's Image Processing Toolbox, but this isn't anticipated to be a significant roadblock. MATLAB is therefore given a 9.
 - (b) **C**: Project members have little experience with C. However, several team members, especially those completing computer science minors, have experience with C++, which is an extension of C. Project members do *not* have experience with any C-compatible image analysis libraries like OpenCV or ImageMagick. C is given a 3.
 - (c) **C++**: Team members participating in computer science minors have experience with C++. However, again, the team has little to no experience using C++ libraries like OpenCV. C++ is given a 5.
 - (d) **Python**: Several team members have experience using Python. However, no one has experience using Python image processing libraries like PIL or Python. Python is given a 5.

- Ease of Use

- (a) MATLAB: MATLAB was designed to be very user-friendly so as to get the programming aspect out of the way and put the math and modeling MATLAB specializes in front and center. It has a highly-developed GUI that makes using its tools straight-forward. MATLAB is very well-documented and has an active community available for consultation. Considering this, MATLAB is given a 10.
- (b) C: Relative to MATLAB, C is a lower level language. Operating on matrices is more difficult, doing higher-level modeling can be more difficult, debugging code and getting useful feedback can be more difficult, and so on. C is given a 5.
- (c) C++: C++, as an extension of C, has many features that C lacks. However, at its core, it's the same platform and operates in a very similar way. It seems reasonable to give it one higher mark than C received as it's easier to use, but not by any great leap.
- (d) Python: Python intentionally uses a very intuitive syntax, which certainly adds to its ease of use. However, like C and C++, it isn't explicitly designed for things like visualizing and modeling. It's easier to operate on matrices in Python than in C/C++, but still harder than it is in MATLAB. Python receives a 7 in this category.

- Suitability

- (a) MATLAB: MATLAB is naturally designed to easily handle matrices, which will most likely be the basis for the format of the data provided by the sensor system. MATLAB has extremely useful built-in tools like its Image Processing Toolbox. It's also an excellent option for visualizing data and otherwise getting feedback on how correct code is and how accurate the models being used. MATLAB has no drawbacks in this category, and is hence given a 10.
- (b) C: C isn't as strong at working with matrices and lacks the advanced visualization features that MATLAB has. C has a number of freely-available, widely-used image processing libraries like OpenCV and ImageMagick. The OpenCV library, though written in C++, *can* be used in combination with C. Last year's project KESSLER using the OpenCV library themselves. While using C would likely be more difficult than using MATLAB, there is strong precedent for using it for this type of data processing. To elaborate on OpenCV: it comes prepackaged with a wide variety of algorithms that largely focus on identifying objects in images, constructing virtual models of objects it identifies, and so on. These are exactly the capabilities that would be useful in constructing the sensor data processing subsystem. C is thus given a 8; it can leverage libraries that are likely of similar usefulness, though it isn't as adept at manipulating matrices.
- (c) C++: C++ is an extension to C and in general, libraries available for C are also available for C++. OpenCV library, again, is widely accepted as an excellent visual processing library. It's similarly adept at manipulating matrices as C is. Because OpenCV was written in C++ and was originally intended to be used in that language, it seems reasonable that the team might have fewer difficulties working with it if C++ is chosen as the development language. C++ is therefore given an 9, slightly higher than C was given.
- (d) Python: Python has a wide variety of image processing libraries available, including PIL (Python Imaging Library), Pillow, and a wrapper for OpenCV. Of these three, OpenCV is likely the most promising option as PIL and Pillow largely focus on image manipulation and editing rather than being able to intelligently detect objects within images. Python handles multidimensional arrays in a more dynamic way (in terms of being able to create, edit, shrink, or expand them dynamically) than C/C++. However, that doesn't seem like a particularly significant consideration, and thus Python is given the same score as C++ - an 9.

- Convertibility

- (a) MATLAB: MathWorks has developed a tool called MATLAB Coder designed to convert MATLAB code to C/C++ code. The Image Processing Toolbox is described as being largely compatible with MATLAB Coder. For this reason, MATLAB is given an 8.
- (b) C: If C is chosen as the development language, it will almost certainly be chosen as the operational language as well as it has very good performance, thereby negating the need for any conversion. C is given the highest score.
- (c) C++: As above, if C++ is chosen as the development language, it will likely stick around as the operational choice, again negating the need for any code conversion. C++ is also given the highest score.

- (d) Python: There are a variety of compilers capable of converting Python code to C/C++; one example is Cython. Such libraries seem well-developed and widely used. Thus, Python is given an 8.

- Cost

- (a) C, C++, and Python are all free languages. Furthermore, as University of Colorado students, MATLAB is available to the entire team at no cost. Therefore, all four software options are given the highest possible score.

	Weight	MATLAB	C	C++	Python
Prior Knowledge	15%	9	3	5	5
Ease of Use	25%	10	5	6	7
Suitability	30%	10	8	9	9
Convertibility	20%	8	10	10	8
Cost	10%	10	10	10	10
Weighted Total	100%	9.45	7.1	7.95	7.8

Table 47. Trade Study for Sensor Data Processing Software (Development)

Operational

The following list catalogs the parameters for the operational sensor data processing software trade study and descriptions of why their particular weights were chosen.

- Prior knowledge - Prior knowledge is denoted by the amount of experience the software subteam members have with a programming platform. A large amount of prior knowledge of the software being considered will make it easier to develop in. However, since it is fairly easy to learn how to use new software for operation purposes and the code structure will already have been developed, it is weighted less than ease of use, with a value of 0.10.
- Ease of Use - The score for this parameter is based on how well documented the software is, how easy it is to debug, and how much manufacturer or community support there is for the option. Being strong in all of these factors will help make the sensor data processing subsystem more functional and accurate, hopefully leading to a more refined result. For this reason, ease of use is weighted heavily, with a value of 0.2.
- Speed - This is a simple definition: how fast does this language compile or interpret scripts and execute? Seeing as the purpose of dividing the control systems software into development and operation software is to quickly develop and verify the software in order to then, potentially, convert it into another, more efficient language in order to improve the overall speed of the control loop, this is the most important category for operation. This category will be assigned a weight of 0.4.
- Suitability - Since the development code will likely be in a different language than the operational software, it will be very important to convert to a language that is suited for or tailored to our specific use: robotic systems. If the operational language has more support libraries for our uses, this will be a *major* benefit and add lots of value. This category will be assigned a weight of 0.3.
- Heritage - Reinventing the wheel is an unnecessary waste of time. Where it is compatible with the goals of our sensor data processing subsystem, it should be a goal to reuse as much code from last year's project KESSLER as possible.

	0-2	3-6	7-9	10
Prior Knowledge	None	Introductory to basic	Proficient to extensive	Expert
Ease of Use	No support, difficult read/write	Open-source support	Official support, easier read/write	Easiest read/write
Speed	Slowest	Medium, optimizable	Fast	Fastest
Suitability	Interpreted	Compiled, high-level	Compiled, lightweight	Compiled, most basic

Table 48. Scoring Parameter Scale for Sensor Data Processing Software Choices (Operational)

- Prior Knowledge
 - (a) Refer to the section above table 47; team members have significant experience with MATLAB, less experience with C++ and Python, and even less experience with C. The operational scores for prior knowledge given in table 49 are the same as those given in for the development version of the system in table 47.
- Ease of Use
 - (a) Again, refer to the section above table 47, where the ease of use of MATLAB, C, C++, and Python are described. The ease of use during development and operation have no reason to be different, and thus the scores given in table 47 are reused in table 49.
- Speed
 - (a) MATLAB: MATLAB is the slowest language by far. It is interpreted rather than compiled. It receives a 2.
 - (b) C: C is the fastest language, marginally faster than C++. It receives a 10.
 - (c) C++: C++ trades off its speed for ease of use, coming in just slower than C when features exclusive to C++ (not present in C) are used. It receives an 9.
 - (d) Python: Python is interpreted and still slower than C/C++, but faster than MATLAB and can be optimized for speed in different areas of use with its specific support libraries. It receives a 5.
- Suitability
 - (a) MATLAB: MATLAB is an interpreted language, lending little to its strength in operations by design. MATLAB cannot run on embedded systems, therefore demanding that the grappling arm's "brains" be powered by a personal computer. MATLAB receives a 2.
 - (b) C: C is a lightweight compiled language, and so it is designed for embedded systems like our robotics software and can be tailored for speed and operational efficiency. It receives a 10.
 - (c) C++: C++ is similar to C, however it was designed away from the basics of C slightly so that it could be tailored to more specific purposes as discussed earlier; it is still compiled and lightweight, giving it great operational strength. It receives an 8.
 - (d) Python: Python exhibits aspects of compiled and interpreted languages together, with the ability to be executed in both manners. It could be optimized for the team's purposes and treated as lightweight and compiled, but it is not as efficient as C/C++ for robotic control systems. It receives a 6.
- Heritage
 - (a) MATLAB: Per KESSLER's spring final review, MATLAB was ultimately used for the visual processing of data from the Kinect camera that project utilized. That code isn't currently available, but should it be found, it could prove a great aid in developing MEGACLAW's sensor data processing subsystem. Note that KESSLER aimed to grapple a flat plate, like the body of the spacecraft (then modeled as a downscaled Iridium satellite) or a solar panel. Conversely, this project currently aims to grab an external grapple point of a predefined shape, which would require extensive modifications to KESSLER's image processing code were it to be used. MATLAB is given a score of 7.
 - (b) C: None of KESSLER's code is written directly in C; almost all of it is written in C. Because C++ is derived from C, converting C++ code from KESSLER to C wouldn't be as difficult as it would if C and C++ were completely unrelated. Thus, instead of getting a 0, C will be given a score of 3.
 - (c) C++: Almost the entirety of KESSLER's software is based in C++. ROS (the **R**obot **O**perating **S**ystem) is written in C++. However, KESSLER's spring final review is adamant that their image processing was done in MATLAB - thus, it seems unreasonable to award C++ a higher score than MATLAB. C++ is thus given a 6.
 - (d) Python: None of KESSLER's image processing code was written in Python, which is thus given a 0.

	Weight	MATLAB	C	C++	Python
Prior Knowledge	10%	9	3	5	5
Ease of Use	20%	10	5	6	7
Speed	35%	2	10	9	5
Suitability	25%	2	10	9	6
Heritage	10%	7	3	6	0
Weighted Total	100%	4.8	7.6	7.7	5.15

Table 49. Trade Study for Sensor Data Processing Software (Operational)

5.2.2. Control Software

Development

The following table lays out how certain capabilities are awarded certain scores in the trade study for the development portion of the control software subsystem. In general, software choices for the control system development were judged on the same scale as were the software choices for sensor data processing development (see: immediately above table 46). However, an option's *suitability* and *convertibility* were weighted differently for the control subsystem:

- Suitability - This is defined as the amount of available functionality the option has with respect to the development, simulation, and optimization of control systems. Having a higher suitability score would result in having more options and flexibility in designing the control system. High suitability also contributes to ease of use because having a lot of already available functionality means we'll have to write less code by hand. For these reasons, suitability has a fifth of the trade's weight, at a value of 0.2.
- Convertibility - It's natural that a model of a control system constructed for development purposes likely can't be directly applied to the actual system. Thus, it's of paramount importance that the model can be converted to a platform that can be directly used by the final grappling arm, and convertibility is thus weighted as 0.3.

	0-2	3-6	7-9	10
Prior Knowledge	None	Introductory to basic	Proficient to extensive	Expert
Ease of Use	No support, difficult read/write	Open-source support	Official support, easier read/write	Easiest read/write
Suitability	No built in control system development functionality	Has control system modeling and simulation functionality	Has control system modeling, simulation, and optimization functionality	Has a wide variety of the aforementioned functionality
Convertibility	Impossible	Manual conversion required	Automatic conversion capability	No conversion needed
Cost	Prohibitive	Expensive	Cheap	Free

Table 50. Scoring Parameter Scale for Control Software Development

The following list contains the reasoning and justification behind why the specific scores were given to each option.

- Prior Knowledge
 - (a) MATLAB and Simulink: The entire team is very familiar with MATLAB, but none of the team has experience with Simulink. For this reason, they get an 8 in prior knowledge
 - (b) MSC Systems and Control: No one on the team has ever been exposed to this software, so it gets a 0.
 - (c) OpenModelica: No one on the team has been exposed to this particular software, but it is built in C, which much of the software team has some experience in, so OpenModelica gets a 5 in prior knowledge.
- Ease of Use
 - (a) MATLAB and Simulink: MATLAB and Simulink have a wealth of documentation and built in functionality that makes it very easy to debug. So MATLAB and Simulink get a 10.

- (b) MSC Systems and Control: This option has a high level of support from the manufacturer and decent documentation. However, this option has a score of 7 in ease of use because there is not much community support for it.
- (c) OpenModelica: OpenModelica has a good amount community documentation, but some of it may be out of date or unreliable. Additionally, it's based on C, which is notoriously hard to debug. For these reasons, it is given a 3 in ease of use.

- Suitability

- (a) MATLAB and Simulink: Simulink has a wealth of built in functionality as well as compatible external libraries that assist in modeling, simulating, and optimizing control systems. For this reason, it gets a 10 in suitability.
- (b) MSC Systems and Control: This software is built to model and simulate dynamical and control systems, but it lacks functionality for analyzing and optimizing control systems. For this reason it gets a 6 in suitability.
- (c) OpenModelica: OpenModelica has a large range of functionality when it comes to modeling, simulating, and optimizing control systems. However, since it is less widely used than MATLAB and open source, there is not as large a variety of available functions. For this reason OpenModelica is given an 8 in suitability.

- Convertibility

- (a) MATLAB and Simulink: MATLAB's has a supporting library which can convert existing MATLAB code into C and C++, giving this option automatic conversion capability if the operational software is chosen to be C or C++. For this reason, MATLAB and Simulink get a score of 8.
- (b) MSC Systems and Control: There is no apparent way to convert the models generated in this software to another language, so it gets a score of 0.
- (c) OpenModelica: The models generated in this software come in the form of C code by default. This means that if C is chosen as the operational language, no conversion would be necessary, so this option gets a 10.

- Cost

- (a) MATLAB and Simulink: The whole team already has licences for MATLAB, so it gets a 10 in cost.
- (b) MSC Systems and Control: This option is designed for use in industry, and it requires getting a quote to find the actual price of the software. For this reason, it is safe to say that this option would be prohibitively expensive and is given a 2 in cost.
- (c) OpenModelica: This is free, so it gets a 10.

	Weight	MATLAB and Simulink	MSC	OpenModelica
Prior Knowledge	15%	8	0	5
Ease of Use	25%	10	7	3
Suitability	20%	10	6	8
Convertibility	30%	8	0	10
Cost	10%	10	2	10
Weighted Total	100%	9.1	3.15	7.1

Table 51. Trade Study for Control Software (Development)

Operational

The weights for each criteria considered in the trade study for the operational control software subsystem are weighted in a similar manner as the weights for the operational sensor data processing subsystem (see: immediately above table 48). However, last year's project KESSLER operated in an open-loop fashion, there is no heritage to consider for the

control system. Hence, *heritage*, which was weighted as 0.1, was removed, and the weights of the two most important parameters, speed and criteria, were increased by 0.05 to 0.4 and 0.3 respectively.

As before, the following table lays out how certain capabilities are awarded certain scores for each of the categories defined above.

	0-2	3-6	7-9	10
Prior Knowledge	None	Introductory to basic	Proficient to extensive	Expert
Ease of Use	No support, difficult read/write	Open-source support	Official support, easier read/write	Easiest read/write
Speed	Slowest	Medium, can be optimized	Fast	Fastest
Suitability	Interpreted	Compiled, high-level	Compiled, lightweight	Compiled, most basic

Table 52. Scoring Parameter Scale for Control Software (Operational)

Now, after having assigned weights and scales to the characteristics of these different operational programming languages, each language in consideration must be measured against these metrics in a trade study to determine which will be the best fit. Below is a breakdown of each language and the scores assigned to it in each category.

- Prior knowledge -
 - (a) MATLAB: On average, every member of the software subteam has extensive prior knowledge with MATLAB. It receives a 9.
 - (b) C: A few members of the software subteam have basic knowledge regarding C from previous coursework and projects. It receives a 5.
 - (c) C++: Every member of the team took some coursework involving C++ and some have used it for multiple projects, so the prior knowledge of the team is just above basic, slightly differentiation it from C. It receives a 6.
 - (d) Python: The same is true for Python as is for C++. It receives a 6.
- Ease of Use -
 - (a) MATLAB: MATLAB is the easiest to use, read and write of all of these languages, and it provides plenty of official support. It receives a 10.
 - (b) C: C is the most difficult language to read and write in, but it comes with plenty of open-source support and little official support. It receives a 4.
 - (c) C++: C++ is more user-friendly than C, with more official support and easier read and write . It receives a 7
 - (d) Python: Python is just below MATLAB in readability and ease of writing, and there are thousands of available support libraries for Python as well as a fair amount of official support. It receives a 9.
- Speed -
 - (a) MATLAB: MATLAB is the slowest language by far. It receives a 1.
 - (b) C: C is the fastest language, marginally faster than C++. It receives a 10.
 - (c) C++: C++ trades off its speed for ease of use, coming in just slower than C. It receives an 8.
 - (d) Python: Python is slower than C/C++ but faster than MATLAB and can be optimized for speed in different areas of use with its specific support libraries. It receives a 5.
- Suitability -
 - (a) MATLAB: MATLAB is an interpreted language, lending little to its strength in operations by design. It receives a 2.
 - (b) C: C is a lightweight compiled language, and so it is designed for embedded systems like our robotics software and can be tailored for speed and operational efficiency. It receives a 10.

- (c) C++: C++ is similar to C, however it was designed away from the basics of C slightly so that it could be tailored to more specific purposes as discussed earlier; it is still compiled and lightweight, giving it great operational strength. It receives an 8.
- (d) Python: Python exhibits aspects of compiled and interpreted languages together, with the ability to be executed in both manners. It could be optimized for the team's purposes and treated as lightweight and compiled, but it is not as efficient as C/C++ for robotic control systems. It receives a 6.

	Weight	MATLAB	C	C++	Python
Prior Knowledge	10%	9	5	6	6
Ease of Use	20%	10	4	7	9
Speed	40%	1	10	8	5
Suitability	30%	2	10	8	6
Weighted Total	100%	3.9	8.3	7.6	6.2

Table 53. Trade Study for Control Software Development

So, overall Table 53 concludes via trade study that C is the strongest candidate for use as the team's operational programming language for control software.

5.3. Hardware Trade Studies

5.3.1. Grapple Point Manufacture

In this trade study we will be comparing the manufacturing options for the designed grapple point. As described earlier, these options are 3D printing a designed object, building it out of Lego's, construction using cardboard, and purchase of a pre-made object. The key design effects we will be weighing on are: cost, durability, ease of manufacture, and formability.

- Cost: The cost will be scored based on how much it would cost to make the grappling point and any possible replacements. After some research, it was found that most of the manufacturing options would cost under \$100, so anything over this amount would result in a 0.
- Durability: Durability will be scored based on how likely the grapple point is to break if it were dropped or crushed by the end effector. A score of 10 means it will not drop or break and will not need replacement. All scores below that are explained in the table below.
- Ease of manufacture: This score depends on multiple factors including: the time it takes to create, how easily it can be remade if broken, and the ability to easily update the design.
- Formability: The formability of the material defines the ability to create different shapes. This will be scored in comparison to the five grapple point design options. High points means that material can easily create any of the designs, including curves like for the cylinder.

	0-2	3-6	7-9	10
Cost	Over \$50 to \$100	\$20 - \$50 total	Less than \$20	Free to manufacture
Durability	Unable to support it's own weight or easily breakable if bumped	Will break if dropped and the grappling force could crush the object	Might break or deform if dropped but could withstand the grappling force	Will not break or deform if dropped or a large grappling force is applied
Ease of Manufacture	A design alteration would require complete re-make of the object, would take multiple hours to create	Design alterations difficult but possible, would take time and effort to fix damages	Design alterations would take less than an hour to make, could quickly be rebuilt if damaged	Could easily update the design, requires little time to manufacture
Formability	Cannot create any possible design options (0) One possible grappling point design option (2)	Two grappling point design options (4) Three grappling point design options (6)	Manufacture 4 of the possible grappling point design options (8)	Manufacture all five of the grappling point design options

Table 54. Scoring Parameter Scale for Manufacture of the Grappling Point.

	Weight	3D Printed	Lego's	Cardboard	Purchased Object
Cost	10%	10	5	9	6
Durability	30%	9	5	3	10
Ease of Manufacture	20%	2	10	6	9
Formability	40%	10	6	6	6
Weighted Total	100%	8.1	6.4	5.4	7.8

Table 55. Trade Study for Manufacture of the Grappling Point.

- **3D Printing:** Since the Lulzbot's are free to use at the ITLL, 3D printing received a score 10 for cost. PLA is known for being durable, and would likely be able to withstand any force from the end effector. It received a score of 9 since it might be damaged if dropped. Since it would require finding or designing a .stl file for printing, as well as take multiple hours to print, the ease of manufacture scored at a 2. A 2 was given instead of a 1 or 0 since it is fairly easy to find premade .stl files online. Lastly, it received a score of 10 for formability since any possible design option could be found or created in CAD.
- **Lego's:** After some research, basic lego block sets were found to cost roughly \$20. Since other style of lego's may be needed depending on the design of the grappling point they received a cost score of 5. The durability of Lego's scored a 5. This is because the shape created will fall apart if dropped, but the legos themselves will not break. However, any shape could easily be remade within a matter of minutes which is why the ease of manufacture score is a 10. Finally, for formability Lego's scored a 6. This is because their ability to form the ring or the cylinder is limited due to the weakness between Lego blocks and the limited availability of curved blocks.
- **Cardboard:** Since cardboard recycling is readily available it received a cost score of 9. A 10 was not earned since it is not certain that all types and sizes of cardboard would be available for free. Cardboard earned a 3 for durability since it would require glue or tape to manufacture the desired grappling point design. These bonds would easily break under pressure and cardboard can easily bend or dent. A score of 6 was received for both ease of manufacture and formability. For ease of manufacture, it would take time to let glue dry and to cut the cardboard into the desired shapes for manufacture, it would also require rebuilding from scratch if the cardboard were to be dented. Cardboard could also only easily build three of the five design options since it cannot easily be curved.
- **Purchased Object:** Purchasing an object would cost roughly between \$20 and \$50 depending on any design updates or replacements needed, therefore it scored a 6 for cost. A score of 10 was given for durability since a purchased object would be kids wooden or plastic shapes which are known for being able to withstand being dropped. Since purchasing an object required no manufacturing a 9 was received for ease of manufacture. It did not get a 10 just because if a replacement was needed, the right object would still need to be found and purchased. Lastly, a score of 6 was given for formability since it would be difficult to find a knob or ring toy at the desired dimensions.

5.3.2. Grapple Point/Object Design

In determining the grapple point design, the different designs discussed in the section for Key Design Options Considered were ran through a trade study to determine the best option for the project. The design parameters are shown in table 56.

	0-2	3-6	7-9	10
Ability to Track	High difficulty for sensor tracking	Low difficulty for sensor tracking	High capability of tracking for sensor	Stand alone object easy for sensors to track
Gripping Capability	Low end-effector grappling capability	Limited end-effector grappling capability	Majority end-effector grappling capability	All end-effector grappling capability
Ease of Manufacturing	Extremely difficult to manufacture	Somewhat difficult to manufacture	Somewhat easy to manufacture	Easy to manufacture and recreate
Realism	No equivalence to satellites	Relative to a limited amount of satellites	Relative to a majority of satellites	Fully representative of a part on a satellite

Table 56. Scoring Parameter Scale for Grapple Point Design

- **Ability to Track:** The tracking capability was chosen for the trade study as it is important for the sensors to be able to identify the grapple point quickly. Designing the grapple point to be easily seen allows for the grappling sequence to be done in the allotted time sequence. Since the identification of the grapple point is vital for the mission, the tracking capability for the trade study was given a weight of 35%.
 - (a) Flat Plate: The flat plate was given a value of 6 since the satellite will be composed of mainly flat surfaces, so it might be difficult to track at first.
 - (b) Cylinder: The cylinder was given a value of 8 since it is easily identifiable, but is similar to rods that might be placed on the satellite.
 - (c) Knob: The knob was given a value of 9 since it is unique to satellites and would be very easily identifiable.
 - (d) Ring: The ring was given a value of 8 since it is a unique design and easily identifiable, but the ring may be too small to sense.
 - (e) Cube: The cube was given a value of 9 since it is unique to satellites and can be easily identifiable whilst sensing.
- **Gripping Capability:** Gripping capability is the most important part for designing a grapple point. The grappling point needs to be designed such that it is able to be grappled easily by the end-effector and that the end-effector does not have to go through too many orientation changes just to be able to grapple it. Therefore, the gripping capability was chosen to have a weight of 45% in the trade study.
 - (a) Flat Plate: The flat plate was given a value of 8 since it has a great surface area and is capable of being grappled by a majority of end-effector's. However, the flat plate is only allotted one orientation for grappling, making it more difficult for the arm.
 - (b) Cylinder: The cylinder was given a value of 2 since it is not optimal for a majority of the end-effector's. It contains a small surface area and is really only capable of grappling by the appendages end-effectors.
 - (c) Knob: The knob was given a value of 1 since it is not optimal for any end-effector design. It also contains a low surface area.
 - (d) Ring: The ring was given a value of 3 since it is optimal for the hook end-effector, but is otherwise useless to the other end-effector's.
 - (e) Cube: The cube was given a value of 7 since it is capable of being grappled by a majority of the end-effectors, but is large in volume leading to less margin of error.
- **Ease of Manufacturing:** The ease of manufacturing is not as important while designing the grapple point. However, it is important that the grapple point can be readily manufactured again if there is a problem with the grapple point. The trade study for grapple point manufacturing specifies that the optimal manufacturing technique is 3D printing which is readily accessible and free for students. Due to these criteria, the weight of the ease of manufacturing for the trade study was chosen to be 15%.
 - (a) Flat Plate: The flat plate was given a value of 10 since it is easy to manufacture in all methods of manufacturing.
 - (b) Cylinder: The cylinder was given a value of 5 since it is not easy to manufacture in all methods of manufacturing.

- (c) Knob: The knob was given a value of 0 since it is highly difficult to manufacture this design in all methods of manufacturing.
 - (d) Ring: The ring was given a value of 1 since it is difficult to manufacture in all methods of manufacturing.
 - (e) Cube: The cube was given a value of 4 since it is difficult in a majority of methods of manufacturing.
- Realism: The realism aspect of the grapple point is to describe if the grapple point is of a shape that would be commonly seen on an actual satellite. The customer specified that the realism of the grapple point was not scoped for this project, therefore, the weight of the realism in the trade study was chosen to be 5%.
 - (a) Flat Plate: The flat plate was given a value of 10 since it is representative to that of a solar array.
 - (b) Cylinder: The cylinder was given a value of 8 since it is representative to that of a rod on a satellite.
 - (c) Knob: The knob was given a value of 0 since it is not representative of any structure on a satellite.
 - (d) Ring: The ring was given a value of 1 since it is not likely to be on a satellite structure.
 - (e) Cube: The cube was given a value of 4 since it is not common to have a cube on a satellite.

	Weight	Flat Plate	Cylinder	Knob	Ring	Cube
Ability to Track	35%	6	8	9	8	9
Gripping Capability	45%	8	2	1	3	7
Ease of Manufacturing	15%	10	5	4	6	6
Realism	5%	10	8	0	1	4
Weighted Total	100%	7.7	4.85	4.2	5.1	7.4

Table 57. Trade Study for Optimal Grapple Point Design

5.3.3. Robotic Arm Design

- Range of Motion: The range of motion is an important variable to consider because the arm must be able to maneuver to the grapple point while avoiding potential obstacles. With a higher range of motion, the arm should be able to move around different obstacles that may be on a satellite. A higher range of motion would also enable the arm to grapple in more orientations. The range of motion of the arm is directly correlated to how many joints are present in the design. Therefore, the number of joints will be the measure for the range of motion. Because range of motion is so important to the actuation of the arm and its ability to follow the directed path, it is weighted with 0.3 (30%).
- Servo Risk: It has been shown in KESSLER's project that the CrustCrawler servos are prone to failure and overheating. This risk grows as more servos and weight is added to the arm. Servo failure is costly in both time and money. If a servo fails, redesign of the arm will probably be necessary. The measure of risk is calculated by finding the maximum torque possible at each servo joint, and comparing that maximum to the stall torque of the joint. The maximum torque at each joint is found by calculating the weight of the following components multiplied by the length. This calculation assumes that all the weight is acting at the total length of the arm section. This may not be an accurate assumption as the arm component is realistically a beam with complex loads, but it overestimates the maximum torque, which is allowable for failure predictions. The quantity for this measure will be $\Delta T = T_{max} - T_{stall}$. The smallest ΔT will be considered for each design. Because the risk of the servos failing is very important to consider due to the costly results of failure, it is weighted with 0.35 (35%). For reference, Design 1 had $\Delta T = 2.18Nm$, Design 2 had $\Delta T = 1.93Nm$, and Design 3 had $\Delta T = 1.93Nm$.
- Complexity: The complexity of the arm is very important for the software problem. As more degrees of freedom are added to the arm, more joints are needed to be considered within the closed loop software. This is a trade off with range of motion. As complexity increases, so does range of motion. The complexity is very important as the closed loop software is the ultimate goal of this project, giving it a weight of 0.25 (25%).
- Cost: The cost of the arm is important to consider as we are limited to a budget. However, since most of the hardware is already bought from previous years, we do not need to be worried about reaching our \$5,000, giving cost a weight of 0.1 (10%).

	0	1	4	7	10
Range of Motion	0 joints	3 joints	5 joints	7 joints	9 joints
Servo Risk	$\Delta T < 0.5 \text{ Nm}$	$\Delta T > 0.75 \text{ Nm}$	$\Delta T > 1.5 \text{ Nm}$	$\Delta T > 2.25 \text{ Nm}$	$\Delta T > 3.5 \text{ Nm}$
Complexity	9 joints	7 joints	5 joints	3 joints	0 joints
Cost	<\$5000	<\$4000	<\$3000	<\$2000	<\$1000

Table 58. Scoring Parameter Scale for Grappling Arm Design

	Weight	Design 1	Design 2	Design 3
Range of Motion	30%	4	6	7
Servo Risk	35%	6	5	5
Complexity	25%	5	4	3
Cost	10%	5	4	2
Weighted Total	100%	5.05	4.95	4.8

Table 59. Trade Study for Design of the Grappling Arm

5.3.4. End Effector Design

- **Complexity:** The complexity of an end effector for this project's purposes can be measured by the number of mechanical apparatuses the end effector uses to accomplish the target grapple. This will be an important metric as the more complex an end effector design is the higher the risk of mechanical failure becomes for that design option. For this reason the complexity of the end effector carries a weight of 0.3 out of 1.
- **Grip:** The gripping ability of an end effector characterizes how well an end effector can grapple a target and maintain the grapple of a target. Strength as well as dexterity are considered when allocating this score. Because the main focus of this project is to control the arm to the target, but not necessarily maintain a strong grip on the target this is not as important of a design aspect. For these reasons grip of the end effector carries a weight of 0.1 out of 1.
- **Compatibility (with Grapple Point):** The compatibility of the end effector characterizes how well the end effector can interface with the Grapple Point design. An end effector which can interface with many different grapple points will score highly in this category. Because this project is also able to design the grapple point this is not a key aspect of design. For this reason the compatibility of the end effector carries a weight of 0.2 out of 1.
- **Adaptability (with Software):** The adaptability of the end effector characterizes the inherent ability of the end effector to be integrated into the project system as a whole. Specifically, the ease of use as utilized in software development is heavily considered. As the main drive of this project is to solve the problem of closed loop robotic arm control this is a key design aspect. Per the direction of the software team it is ideal to have a lower number of moving parts on an end effector therefore this quality is closely tied to the complexity design aspect outlined above. For these reasons adaptability of the end effector carries a weight of 0.35 out of 1.
- **Cost:** Cost of the end effector associates a rough dollar amount estimate to each of the end effector design options. As much of the hardware for this project has already been acquired cost is not a key design aspect. For this reason the cost of the end effector carries a weight of 0.05 out of 1.

	0	1-3	4-6	7-9	10
Complexity	Failure imminent due to complexity	Many moving parts; high chance of failure	Several moving parts; moderate chance of failure	Few moving parts; low chance of failure	No moving parts; no chance of failure
Grip	Unable to grip target	High difficulty in gripping target	Moderate difficulty in gripping target	Low difficulty in gripping target	No difficulty in gripping target
Compatibility	Interfaces with 0 grapple points	Interfaces with 1 grapple point option	Interfaces with 2 grapple point options	Interfaces with 3 grapple point options	Interfaces with all grapple point options
Adaptability	Impossible to track/sense and/or control	High difficulty in tracking/sensing and/or control	Moderate difficulty in tracking/sensing and/or control	Low difficulty in tracking/sensing and/or control	Tracking/sensing and control are trivially simple
Cost	Outside of budget	1000	750	250	50

Table 60. Scoring Parameter Scale for End Effector Design

	Weight	Claw (Clamp)	Multiple Appendages	Interlacing Appendages	Closing Hook
Complexity	30%	8	3	4	9
Grip	10%	8	8	4	2
Compatibility	20%	7	8	4	1
Adaptability	35%	8	4	5	7
Cost	5%	7	4	4	8
Weighted Total	100%	7.75	4.9	4.35	5.9

Table 61. Trade Study for End Effector Design

6. Selection of Baseline Design

The previous section lays out the results for each trade study performed. The following section contains descriptions of these results as well as an overview of the baseline design. In terms of hardware, MEGACLAW consists of a robotic arm, end effector, sensor system, grapple point, and test stand. Following is a description of the choice for each of these components.

6.1. Sensor Location and Hardware Baseline Choices

The optimal sensor location was determined to be offset from the arm. This allows for the best possible view of both the grapple point and the end effector, as the only time it will be obscured is when the arm is very close to the grapple point. This problem is only solved by having a camera at the end effector, but this is not worth the loss in manufacturing and interference. Additionally, the offset doesn't restrict the arm's motion in any way and also requires very little manufacturing to install the sensor. These advantages are also shared with the base, but the base has a slightly worse FOV as it can be more obscured by the arm.

The optimal sensor hardware was found to be the RealSense depth-D435. This sensor operates in the optimal range defined by the project, it has many options available for interfacing with the software choices of this project, and it has the lowest cost of any of the sensor types besides the Kinect. The documentation on this sensor is also impressive, only being beaten by the Kinect which has all of this project's heritage. The only major drawback is in the lighting requirements, but it is still operational given proper use of the IR camera in a low light environment.

6.2. Software Baseline Choices

The software component of MEGACLAW will consist of two high-level subsystems: the sensor processing and the controls suite. Both the estimator and the controls suite will have a development and operational phase to allow for fast iteration and flexibility. The sensor data processing subsystem takes raw data from the sensor as input and determines the states (currently defined as relative positions and orientations) of the grappling point on the target and of the end effector. Tables 46 and 49 present the results of the trade studies on the most appropriate software platform to utilize. They indicate that during the *development* of the subsystem, **MATLAB** is the best choice due primarily to its

great ease-of-use, the team's extensive experience with it, its strong supporting libraries, and its strong visualization capabilities. When constructing the final, high-performance, *operational* version of the subsystem meant to work together with every other subsystem, C++ was evaluated to be the best choice due to its speed, its supporting libraries, team members' prior knowledge of it, and MATLAB's purported ability to convert its own code to C++.

The controls suite consists of a path planner to satisfy **FR 8** and a control algorithm to satisfy **FR 9**. To develop the control suite algorithms MATLAB and Simulink will be used. The main factors leading to this decision are MATLAB's ease of use, suitability and convertibility. MATLAB and Simulink have a wealth of user support and available functions for modeling, simulating, and optimizing control systems adding to both its ease of use and suitability. It has high convertibility since the MATLAB Coder has the capability of converting directly to C code, which is what will be used for the operational control software.

So, once the control suite has been developed and a controls algorithm has been successfully implemented and tested with the arm and grapple point, it's speed will be addressed. The control software development language was chosen entirely to ensure simplicity and ease for the user in order to satisfy **FR 8** and **FR 9**. The control software operational language will simply optimize the MATLAB-developed path planning and control algorithm process in a language tailored for speed and efficiency. As such, the speed, ease of use, prior knowledge, and suitability categories were all considered for the candidates MATLAB, C, C++, and Python in Table 53, and C was chosen as the strongest candidate for an operational controls language. As the most basic, low-level language, and paired with the MATLAB Coder program, C presents itself as the best language option for meeting the control frequency requirement laid out in **DR 1.1**.

6.3. Hardware Baseline Choices

The hardware components of MEGACLAW was based off of the grapple point manufacturing, grapple point design, arm design, and end-effector design.

The grapple point will be manufactured using the 3D printers. Although this will take the longest amount of time to manufacture, it is the most durable and also will be free to manufacture. Since the 3D printers are readily available in the ITLL if any replacements were needed they could be made within the day.

The grapple point design will be designed to be a flat plate. The flat plate ultimately dominated the trade study. While the flat plate ranked the least in tracking capabilities, it is still had a reasonable value for the feasibility of the project. The flat plate provides an easy solution for the end-effector design, as it contains a high surface area for the claw to close on. The flat plate is also optimal for the claw end-effector as it enable the pressure sensors to be utilized in the determination of the grappling and release in order to satisfy **FR 7**.

The claw (clamp) end effector design was selected as the optimal end effector design due to its high compatibility and adaptability and low complexity. The claw (clamp design) will maximize opportunities for successful grapple as it can interface with almost any grapple point design utilizing a relatively high grip strength. Additionally, the low complexity of the end effector design allows for easier integration with software development and minimizes mechanical failure risk. The approximate cost of the claw (clamp) end effector is well within budget if it is necessary to replace the KESSLER heritage end effector.

References

- [1] NASA: *What is Orbital Debris?*
<https://www.nasa.gov/audience/forstudents/5-8/features/nasa-knows/what-is-orbital-debris-58.html>
- [2] Tech Times: *ESA Calls for Sustainable Future in Space, Cites Growing Problem of Orbital Debris*
<https://www.techtimes.com/articles/205654/20170423/esa-calls-for-sustainable-future-in-space-cites-growing-problem-of-orbital-debris.html>
- [3] <https://www.upwork.com/hiring/development/c-vs-c-plus-plus/>
- [4] <https://www.activestate.com/blog/2016/09/python-vs-cc-embedded-systems>
- [5] <https://software.intel.com/en-us/realsense/sr300>
- [6] <https://software.intel.com/sites/default/files/managed/0c/ec/realsense-sr300-product-datasheet-rev-1-0.pdf>
- [7] McBride, Andrew, et al. CASCADE Spring Final Review. Aerospace Engineering Sciences Senior Design Projects, University of Colorado Boulder

- [8] Alvarenga, Glenda, et al. KESSLER Spring Final Review. Aerospace Engineering Sciences Senior Design Projects, University of Colorado Boulder
- [9] The New OPAL Performance Series 3D LiDAR. Neptec Technologies, www.neptectechnologies.com/wp-content/uploads/2018/06/OPAL-3-P-Series_-conical-FOV-88-00201-01-06_2018-1.pdf.
- [10] RIEGL VUX-1HA. RIEGL, 2015, www.riegl.com/uploads/tx_pxpriegl/downloads/DataSheet_VUX-1HA_2015-10-06.pdf.
- [11] <https://www.lulzbot.com/store/printers/lulzbot-taz-6>
- [12] <https://www.lulzbot.com/store/filament/polylyte-pla>
- [13] Intel RealSense Depth Camera D435. Intel, click.intel.com/intelr-realsensetm-depth-camera-d435.html.
- [14] 2017, Stimulant. Depth Sensor Shootout. Stimulant, stimulant.com/depth-sensor-shootout-2/.
- [15] Kinect. Wikipedia, Wikimedia Foundation, 18 Sept. 2018, en.wikipedia.org/wiki/Kinect.
- [16] <http://www.crustcrawler.com/products/dualgripper/>
- [17] <https://patents.google.com/patent/US5570920A/en>
- [18] <https://patents.google.com/patent/US4921293A/en>