A Commercial Imaging and Robotics System for Satellite Space Servicing

Abdiel Agramonte-Moreno*, Glenda Alvarenga† , Thanh Cong Bui‡, Christopher Choate§, Lauren Darling¶, Sergey Derevyanko‖, Cassidy Hawthorne**, Abigail Johnson††, Nicholas Thurmes‡‡ and Jannine Vela§§
University of Colorado at Boulder, College of Engineering
Ann and H.J. Smead Aerospace Engineering Sciences

Satellite rendezvous-and-capture is a need in the Low-Earth-Orbit (LEO) environment to reduce risk of high value assets colliding with debris. This paper presents a prototype capture system based on an integrated imaging and robotic control solution using commercial off-the-shelf hardware and open source software. For the imaging system, a Microsoft Kinect V2 is used to collect two and three dimensional images of objects within its field of view and generate location and orientation data of these objects. A seven degree-of-freedom CrustCrawler Robotics modular robotic arm is used to capture the object while accounting for collision avoidance. The Robotics Operating System (ROS) provides the framework to conduct path planning, actuator control execution, and interfaces for the image processing software compatible with the Microsoft Kinect V2. This prototype system operation is demonstrated by identifying and capturing satellite features on a spacecraft model which emulates common components on LEO spacecraft. Two-dimensional color image analysis, and three-dimensional geometric fitting are used to determine the capture location and orientation of the features. This data is then sent to the control software that will determine an optimal path to then execute the motion and capture the satellite.

I. Introduction

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

Sierra Nevada Corporation (SNC), in collaboration with the Ann and H.J. Smead Aerospace Engineering Department undergraduate senior design team (KESSLER) at the University of Colorado at Boulder, seeks to address the orbital debris issue in LEO by developing technologies that demonstrate the capability to remove space debris using a low-cost small satellite with a robotic arm. The concept of this solution has three operational objectives: (1) perform rendezvous with the space debris, (2) identify space debris and capture with the robotic arm, and (3) use one of several methods to de-orbit the space debris or move the debris to a different orbit. The second operational objective can be demonstrated by tailoring specific commercial-off-the-shelf (COTS) hardware and software (open-source and well documented proprietary) which is the focus of this paper.

This paper is organized as follows. Section 2 introduces the technologies available and considered for KESSLER (COTS hardware and open-source & proprietary software). Section 3 describes the selected design for the imaging system of KESSLER and related test results. Section 4 describes the selected design for the robotics system of KESSLER and related test results. Section 5 details the interface of the imaging and robotics system and planned integrated system testing. Finally, Section 6 presents the conclusion.

*AIAA Undergraduate Student Member: 934447
†AIAA Undergraduate Student Member: 934842
‡AIAA Undergraduate Student Member: 902640
§AIAA Undergraduate Student Member: 903784
¶AIAA Undergraduate Student Member: 486382
‖AIAA Undergraduate Student Member: 903279
**AIAA Undergraduate Student Member: 644091
††AIAA Undergraduate Student Member: 934843
‡‡AIAA Undergraduate Student Member: 890939
§§AIAA Undergraduate Student Member: 934845

II. Available Commercial and Open Source Technologies

A. Required Performance

In order to demonstrate viable functionality of a prototype capture system, the following must be achieved: identification of a satellite and grappling features (solar panels and communication antennae) via image processing, determination of (collision-free) grappling approach path(s) to the grappling features, and capture of the grappling features via a multiple degree-of-freedom (DOF) robotic arm. Table 1 defines the various levels of success for the major aspects of this project. Note that the full system position and orientation requirements (post command execution of the robotic arm) are 2 inches and 10 degrees. Planes are defined to be with respect to the mounting of the arm versus the plane of the end-effector to view mounting with respect to the arm's end-effector.

Table 1 Level Definitions

	Identification	**Processing**	**Command Execution**
Level 1	Identify at least two surfaces on the satellite with varying depths in 3D space.	Identify the distance between the closest point of the satellite and the base of the robotic arm (± 4mm).	Demonstrate end-effector can move to closest point and actuate while facing the parallel plane.
Level 2	Identify grappling feature recognition on satellite.	Determine grappling feature location and orientation to within ± 4mm & ± 5° .	Grapple feature in parallel plane within ± 90 ° end-effector roll angle.
Level 3	Identify collision features on target satellite.	Define keep-out zone to within ± 4mm of collision feature surface, and select grappling feature of less collision risk.	Grapple feature in perpendicular plane (demonstrate additional ROM).

B. Viable Solutions

1. Imaging Design Options

The visual processing system is comprised of two primary technical areas: imaging hardware (which addresses the key customer constraint of using a Red Green Blue (RGB) sensor for satellite feature detection), software (which is the means through which the image data is translated into a target location to be captured). The three types of hardware that are considered for this function are the Microsoft Kinect V2 sensor[2], the Orbbec Astra S[3] sensor, and the Intel RealSense SR300[4] sensor. These sensors are all used for motion tracking. While each sensor is similar, the Kinect V2 has the most documentation on-line, and many people use it for tracking purposes. Other than this, the Orbbec Astra S sensor has the same RGB sensors as the Kinect and similar interfacing. The last sensor, the Intel RealSense SR300, has the same RGB sensor, but it has a much more stringent interface that has gone out of service by Intel.

The two software options considered for the visual processing are MATLAB and C++ with OpenCV. MATLAB features an Image Processing toolbox[5] that comes with various built in functions specifically designed for image processing and visual recognition. MATLAB has extensive documentation and community support, furthermore, the entire team is familiar with MATLAB. C++ with OpenCV, while a powerful compiler, does not provide many built in functions for visual processing. It has less supporting documentation than MATLAB, reducing relative accessibility.

2. Robotics Design Options

The robotics system also contains two primary technical aspects (hardware and software). The hardware aspect of the robotic arm system are specifically: the robotic end-effector (which impacts capturing range of motion (ROM)), the girder lengths (which impacts the arm's maximum length), and actuators (which addresses significantly increasing the ROM of the arm to capture at various angles with respect to the base plate). Since the hardware aspect focuses on components of a robotic arm, various robotic arm development kits are considered such as CrustCrawler, LynxMotion,

and OWI which are brands of modular kits that include the necessary hardware to build a custom robotic arm at a low cost.

For the software aspect, the controls system is comprised of two primary technical areas: control software (which is the means to interpret the data from the motor encoders and commanding the motors to actuate), and software integration platform (which is the method by which the control software is executed). The options considered for both of these technical areas were: MATLAB, C++, LABVIEW, and Robot Operating System (ROS). The first three options have software development kits that are compatible with various servo motors but would require a significant amount of development (kinematic modeling) to fully interface with the robotic system. The purpose of ROS is to make components of a robotic system easy to develop and collaborate. ROS has become an industry standard for robotic systems in recent years and its frame work is well accepted within the robotics community. Most high-end robotics companies are now porting their software to ROS due to its significant cut in software/interface development costs and the ability to access it as an open-source platform.

III. Imaging System

A. Hardware and Software Solution

1. Hardware

The Microsoft Kinect V2[2] is the chosen imaging hardware due to the large amount of on-line documentation, accuracy of sensors, and its capability to acquire a 2D RGB image and a 3D RGB point cloud. The 2D image is simply acquired with the RGB camera sensor, which can be seen on the Kinect V2 in Fig. 1 and 2. The 3D RGB point cloud is captured using both the RGB camera and the 3D depth sensor, which contains an IR camera and IR Emitters, as seen in Fig. 1 and 2. To capture a 3D RGB point cloud, the IR system of the Kinect V2 projects a predetermined pattern of IR dots with a known distance between each dot. The Kinect V2 internally calculates the difference in spacing between the received IR pattern to determine where a point is located in 3D space. Each individual point, as seen in relation to the points around it by the IR camera, creates a 3D point cloud. The 2D RGB image is superimposed onto the 3D point cloud to create a 3D RGB point cloud. The 2D image and 3D RGB point cloud are both fed into a visual processing algorithm that outputs a location and orientation in 3D space to capture the satellite.

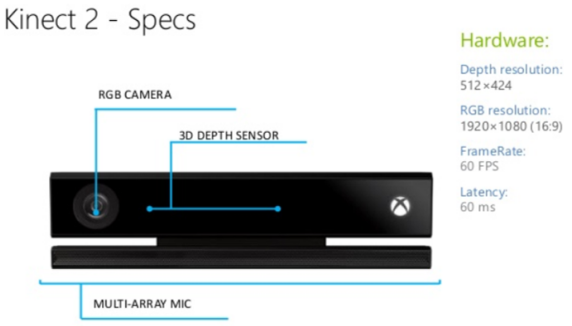

Fig. 1 Specifications of the Kinect V2.

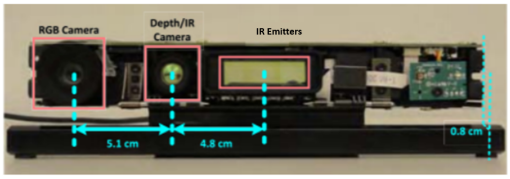

Fig. 2 Internal hardware of the Kinect V2.

In order to verify the accuracy of the processing system it was important to develop a system representative of a real satellite as a test object. A selection trade study occurred that looked at potential satellites by considering: Orbit Altitude, Size, Variety of Features, and Prevalence in operation at the time. Ultimately this indicated that either the Iridium Series or the ORBCOMM FM Series Satellites would be a good target to model. The availability of information on the dimensioning and design of the Iridium Satellite was more readily accessible and was ultimately decided on as the model for the test satellite. The only issue posed by this selection was the satellite was magnitudes larger than the heritage hardware based design. In order to account for this it was decided to scale down the Iridium Satellite for testing to be 30% of each dimension.

Once the size of the satellite was set, the next major step became to determine representative materials and colors. At first the team considered the possibility of using aluminum to model the satellite to give it a level of durability not

available with many other materials while still not posing the danger that suspending a steel structure would pose. Ultimately this was determined to be unnecessary for structural support and was instead substituted for Acrylic panels with High Density Poly-Ethylene rods for the structure design. This allowed the KESSLER team to have a light sturdy structure that would be secure yet also allow for color variation to be applied post manufacturing. The colors selected were gold, silver, and black, three common colors on many satellites in operation. The selection of these colors would allow the KESSLER team to verify the modularity of the Visual Processing System to be later applied to tracking other satellites in future work. The integrated satellite can be seen in Fig. 3.

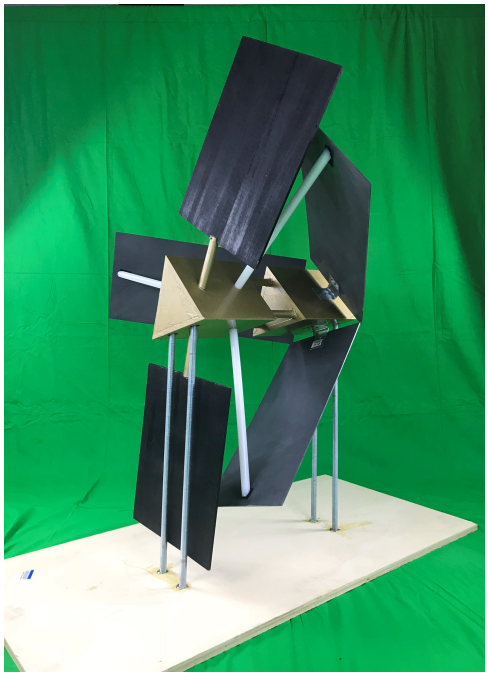

Fig. 3 Test Stand with Scaled Model Iridium Satellite

2. Software

The visual processing algorithm takes in the 2D RGB image and 3D RGB point cloud from the Kinect V2 and finds a location to capture using MATLAB. The algorithm is defined such that the capture location must be on either the solar panels or antennae of the Iridium satellite model. The algorithm begins by identifying that the Iridium satellite is inside the FOV of the Kinect V2. This is done using a feature matching function in MATLAB that compares the 2D RGB image to images in a pre-created database of the satellite model. This database contains images of the Iridium satellite model rotated every 10 degrees about the test stand to provide a complete database. The MATLAB function finds the most unique or drastic pigmentation changes from pixel to pixel in an image to define strong points in the 2D RGB image from the Kinect V2 . The function then compares these points to see if any match the images in the database. Each match is found with 99% confidence. The image in the database with the most matches is identified as the satellite in the FOV and the image analysis continues.

The next step in the algorithm is to identify the features (the solar panels and antennae) based on color. It is defined that the solar panels are black, the antennae are gray, and the bus structure is gold. Using this information and the 2D RGB image from the Kinect V2, the features are isolated based on their color. This is for identification purposes to know what feature is being targeted for capture by the robotic arm. After color-based identification, the 3D RGB point cloud, associated with the 2D RGB image, is used for location analysis. The solar panels and antennas, the target features of the model satellite, can be defined as planes in 3D space. Using a MATLAB function, any planes that are present in the 3D RGB point cloud are defined and isolated. This is completed with a maximum of 3 degrees of error, a constraint that is defined in MATLAB that can be changed. With the planes (either a solar panel or antenna) isolated, the algorithm is finished by determining the point of capture location and orientation.

The capture location is determined by locating a viable capture point on the isolated plane. A viable point is a point

that is on an antenna or solar panel and is within the range of motion of the robotic arm. It also must be close to the edge of the panels so that the claw is able to capture the satellite at that location. The orientation that the robotic arm needs to approach the capture location is also determined from the isolated plane. First a vector is created between the capture location and the center of the plane. Then the angle between the ground and the plane is found. The vector and angle are combined into a quaternion to define the orientation of the capture location. The location and orientation are then sent along to the controls software so that the robotic arm can be actuated to the capture location.

B. Test Results

Various tests have been conducted to validate the accuracy and functionality of the Kinect V2 and visual processing algorithm. The accuracy of the depth sensor, which contains the IR camera and IR emitters, is calculated by taking images of a printed checkerboard with known dimensions at known distances and calculating how the size of the checkerboard varies. This test setup can be seen in Fig. 4 and for the purposes of KESSLER, the checkerboard was placed from 500 to 1400 mm and moved every 50 mm. The error was calculated using MATLAB to be a maximum of 0.1 mm per pixel (which is a relatively small error for a visual system).

Fig. 4 Test setup to find error in the Kinect V2 depth sensor.

The main test for the visual processing algorithm is finding the planes on the satellite model and then locating the closest point on the plane and determining the orientation. Fig. 6 shows a 3D RGB point cloud from the Kinect V2. This point cloud was passed through the visual processing algorithm and output the plane seen in Fig. 7. This plane corresponds to the plane circled in yellow on Fig. 6. It can be seen on Fig. 7 that the closest point is found, in blue, and the center of the plane is found, in red. The vector between the two points defines the orientation of the plane. It should be noted that the error of the closest point on the plane is less than 1 mm and the error of the orientation of the plane is at most 3 degrees.

IV. Robotic System

A. Hardware and Software Solution

1. Robotic Arm Design

The arm that the KESSLER team chose to implement was a COTS modular system developed and distributed by CrustCrawler Robotics[6] which utilizes DYNAMIXEL Smart Servos (a smart actuator system developed by Robotis[7]). The full arm assembly, designed to satisfy the project requirements, is shown in Fig. 8 (left to right: robotic arm base to end-effector). Part of this trade study decision was based on the fact that a previous capstone team had chosen these brands for a similar robotic arm (KESSLER is a partial follow-on project). The heritage hardware was found to be of low risk and was implemented due to the wide variety of safety features and facets that the DYNAMIXEL Servos, and girder options offer. These features include: Inclusive Servo Health Monitoring, Serial Line Communication Protocols, and High Precision Positioning. The girders and associated plates from CrustCrawler were also used for the simplicity of integration.

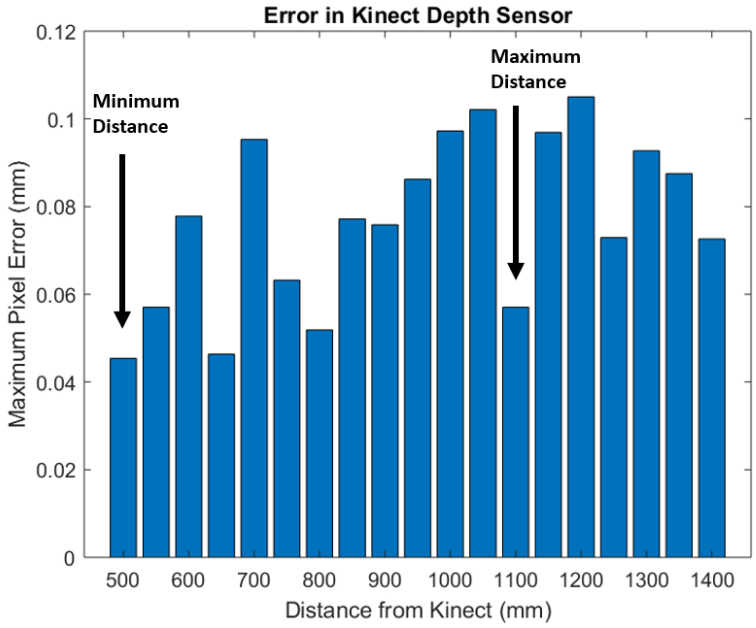

Fig. 5 Mean pixel error of the Kinect V2 depth sensor at various distance from the minimum distance the satellite model can be to past the maximum distance the satellite model will be.

2. Control Software Design

The control software for KESSLER is directed by ROS[8] which is an open-source, industry verified system of packaging and framework for module communication. Additionally it provides tools and libraries for a standard of code development that can be used across multiple platforms and systems. The ability to direct the flow of information uniformly and package it with respect to a community standard significantly increases the robustness and ease of use of our control software.

Overall, the control software runs through the process of: (1) loading in position data, (2) processing that data into a path based on constraints imposed by physical hardware, (3) executes that path to reach the location and orientation data desired, and (4) capture the desired feature. The planned path instructions are built by the MoveIt! path planner, an open-source package distributed through ROS. MoveIt! path planning develops lists of instructions for intermediate steps in the hardware consisting of joint positions and torque values. MoveIt! uses path determination algorithms and control loops used in industry using a setup file that KESSLER has designed in order to model the arm assembly accurately for the processes. In doing this, the following is defined: joints, constraints for operation, collision avoidance constraints and any desired operation modes. Once planned, the path is sent through a trajectory execution manager, processor and specific commanders which route the individual instructions to each appropriate servo. Those packets are sent over a USB UART to be executed by the hardware with the corresponding packet ID. The execution of each packet is watched for incorrect operation by position and torque monitoring functions. Incorrect operation (alarming operations) include overheating, over-torquing, exceeding current limits, etc. During this loop operation this correspondence is always published to a state log. This is a feature of ROS that makes it feasible to set flags for specific operations that should be optimized in the future as well as keep a consistent record of operation.

Once the control software executes the desired path to place the end-effector of the KESSLER robotic arm in an appropriate location, the final capture instructions can be run. These capture instructions are exclusively based on torque constraints of the final grappling actuators. This is done in order to prevent overheating of the actuators. Fig. 9 shows the previous description graphically.

B. Test Results

Controls software testing started with determining motion constraints for the individual actuators. Each actuator is bound to operate within the range appropriate for the hardware it moves. No actuator with girders can operate on all 360 degrees, for instance. In order to verify this behavior, MoveIt! had to be tested to determine to what degree of

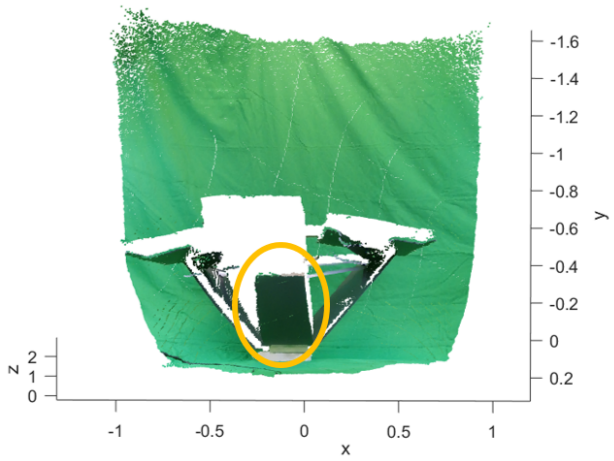

Fig. 6 3D RGB point cloud of the Iridium model.

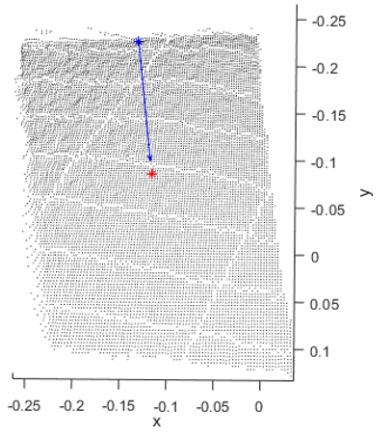

Fig. 7 Isolated plane from the 3D RGB point cloud.

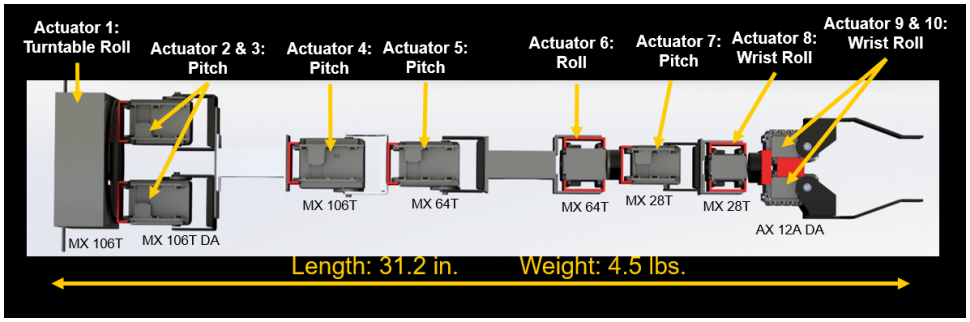

Fig. 8 Full Integrated Arm Assembly

accuracy each path maintained. Fig. 10 below shows the joint position instructions over time in BLUE with the RED lines indicating the upper and lower position bounds. This path is a valid path because none of the blue path instruction to each individual actuator cross the constraint lines.

The next test is the Arm Joint error position test. This test was performed in order to determine if the servos can be: actuated in sync, in a desired time, within a certain position error. The test was done to determine if the position seen by the hardware corresponds with the position desired by the path instruction. The error extrapolated approximately 23 mm of position error, Fig. 11. In this figure the red indicates physical condition of the actuators, the blue indicates desired position of the actuators. This test shows the majority of actuators successfully converge to the desired position within 15 seconds.

V. System Integration

A. Interface Design

Fig. 12 is KESSLER's functional block diagram. The CPU block contains all controlling software and method of peripheral communication. The Hardware block includes all peripherals and physical components of the KESSLER system. Arrows show all connections between the two blocks as well as the designed flow of data. Under the CPU block is exclusively the control and visual processing software. KESSLER is fully integrating data processing, feature recognition, decision making, hardware positioning, and thermal control into a single system. Communication between different aspects is handled by Robot Operating System (ROS). The sensor package is comprised of: a long range visual acquisition device, short range visual acquisition device, and proximity sensor.

The Test Bed includes the physical arm hardware, the test satellite, communication and ground support equipment. The seven DOF arm has both heritage and additional Dynamixel smart servos, and heritage crustcrawler package with

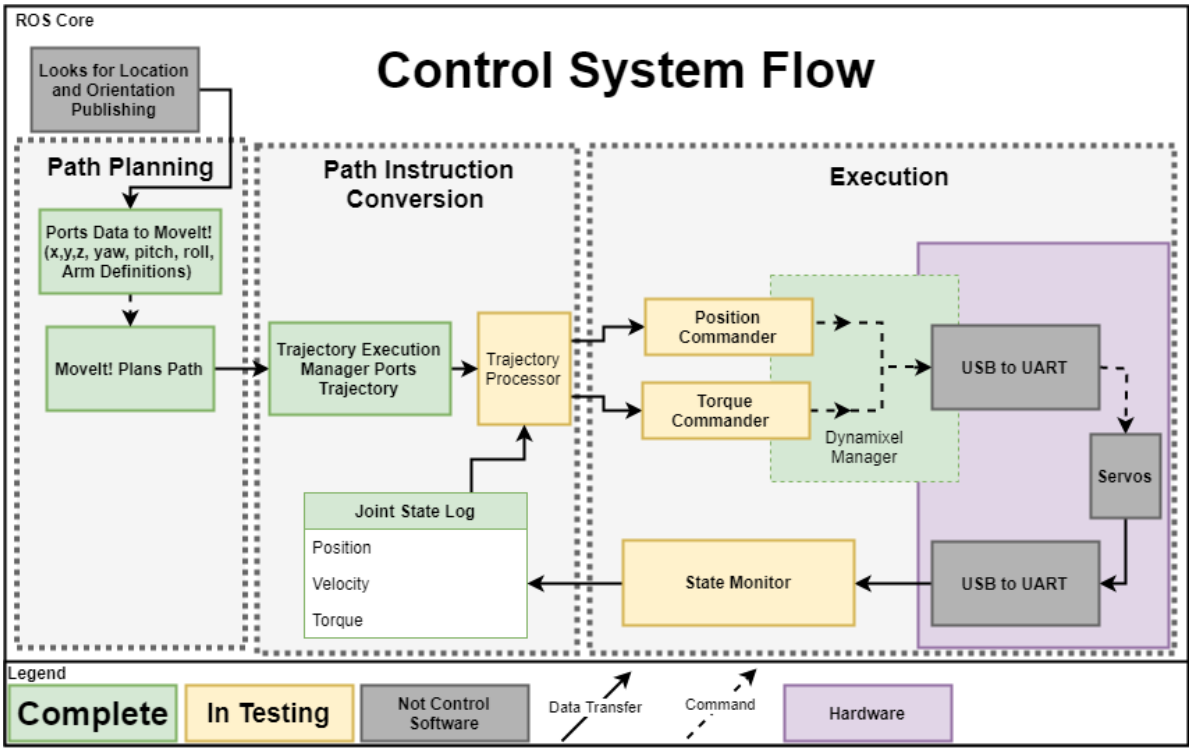

Fig. 9 ROS Control Loop Diagram

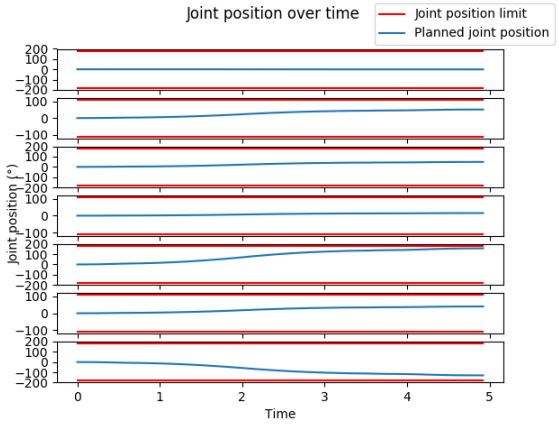

Fig. 10 Motion Constrain Testing Results

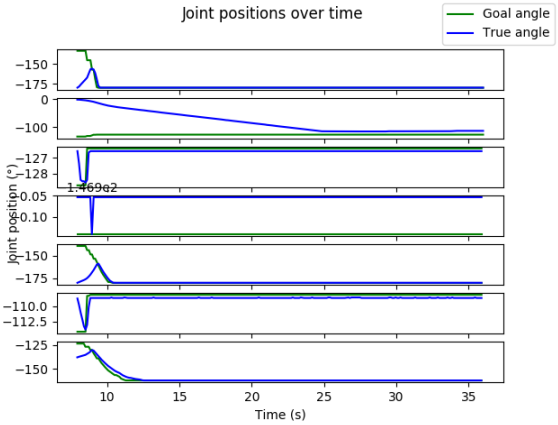

Fig. 11 Position Accuracy Plot over Time

additional designed lengths. Additions of length and servos are designed to extend reach and range of motion. The DYNAMIXEL smart actuator system is designed to be modular with the emphasis on its daisy chain connection scheme. They are a fully packaged Direct Current (DC) Motor with internal gearing, controller, driver and network. Each actuator is individually programmable and network-able, read and monitored through a data packet stream. These are high performance, powerful, versatile devices with supporting documentation and public use software.

The baseline design of the physical hardware can be seen in Fig. 13 which highlights the placement of the simulated satellite, mechanical ground support equipment mounting, the robotic arm, and the imaging hardware. Similarly the software flow can be seen in Figure 14 which provides an overview of how the visual processing and controls software directly interact.

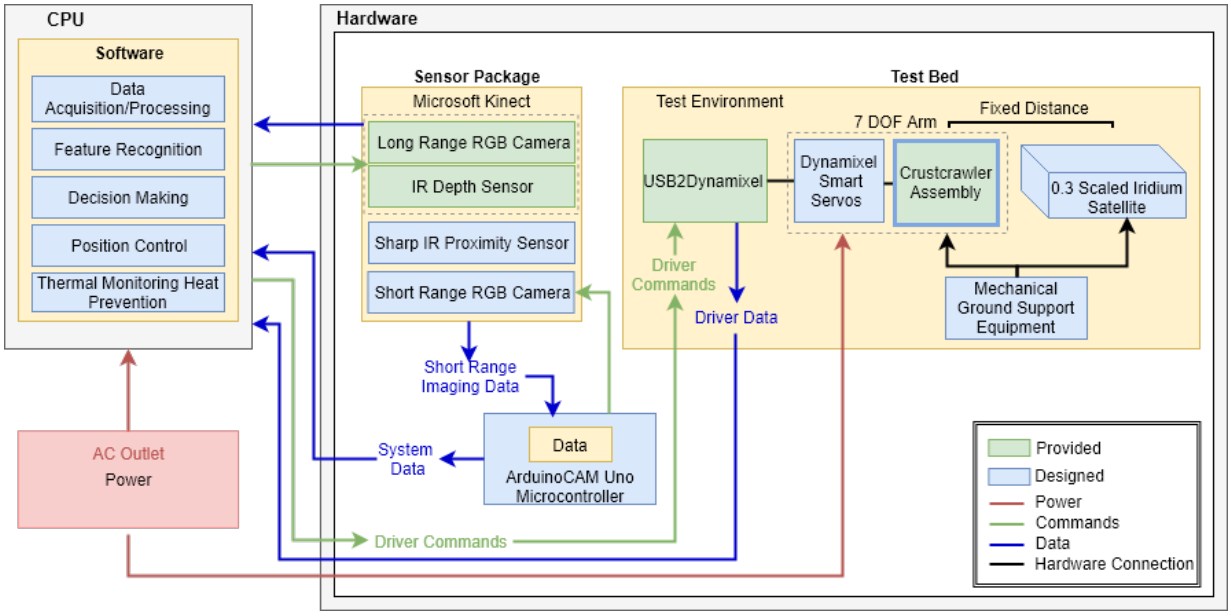

Fig. 12 Functional Block Diagram

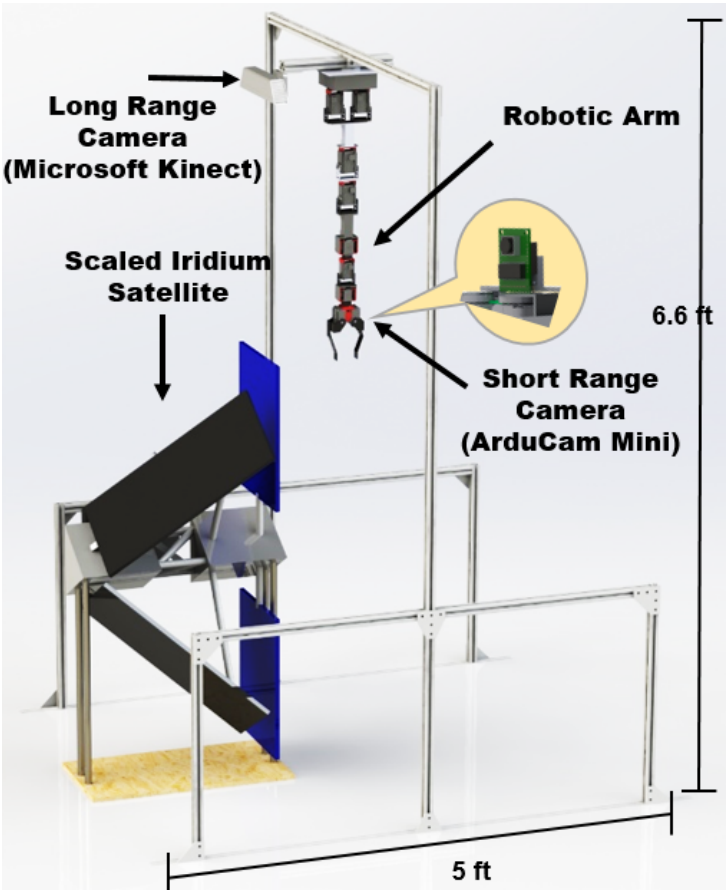

Fig. 13 KESSLER Baseline Design

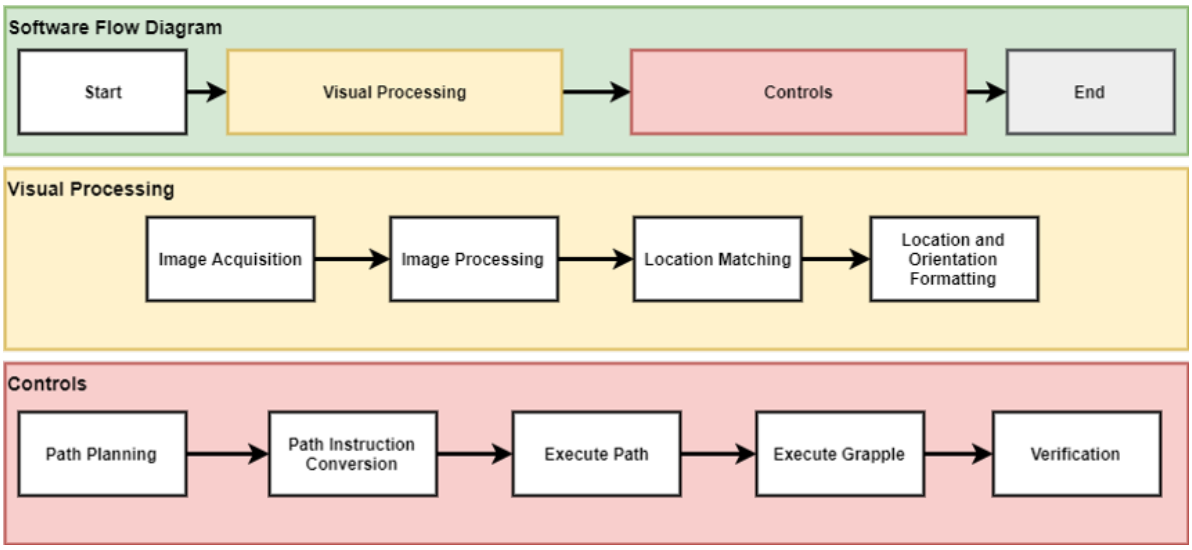

Fig. 14 KESSLER Software Flow

B. Testing

To verify that the final KESSLER system can perform within the requirements, a physical integrated system test must be conducted. In this test the position and orientation of the end effector will be measured using an external sensor to then compare to the commanded position and orientation of the integrated KESSLER system. The primary measure of success will be determined by the difference between the point that the KESSLER system will aim to capture and the true position and orientation of the end effector as measured by another tool when KESSLER completes a given set of operations. The time that KESSLER takes to complete a full operation will also be measured, in addition to the torque produced by the end effector motors and whether the end effector was successful in securing the test satellite.

To measure the true position of the end effector, a Vicon camera system (at the University of Colorado Boulder) will be used. Vicon is capable of returning the position of static objects in three-dimensional space with a precision of 1mm (higher accuracy than KESSLER's requirements). For Vicon to effectively track the end-effector position, motion capture pearl markers are to be mounted on the end-effector.

The KESSLER system has three levels of success (as seen in Table 1). For level one success testing, capturing the satellite is not necessary, though the end effector will actuate upon reaching its target. The test satellite may be moved in between the times that the closest point has been recorded and when the arm is commanded to actuate to that point. This is in order to prevent the arm from colliding with the satellite as it moves to the defined position. For the second level of success, the arm will be required to capture the satellite at the point and orientation that is defined by the visual system. The orientation and position of the end effector after operation will be measured with the Vicon system, and will visually confirm that KESSLER has successfully captured the satellite. For the third level of success, the arm is required to secure the satellite after it has navigated around an obstacle (minimum distance of 2 inches away from surface) that would have prevented the arm from reaching its objective.

The error from the arm is anticipated to be the sum of all the errors that are compilation of both visual processing and control software. The final position of the end effector as measured by VICON is anticipated to be under 0.72 inches away from the the position that is determined by the visual processing algorithm. For levels of success two and three, the final orientation of the end effector is anticipated to have error no more than 3 degrees than the orientation that is commanded by the end effector. These errors fall well below KESSLER's requirements of 2 inches and 10 degrees of error. In addition, the time to conduct a full operation is expected to take less than 3 minutes, which is a sum of the calculations of the visual processing and controls algorithms, and the time it takes the arm to actuate to its target. This is well bellow KESSLER's mission time requirement of 17 minutes. With the completion of these tests, KESSLER shall be capable of verifying its mission requirements.

VI. Conclusion

In conclusion, the KESSLER system is designed to mitigate the growing concern of space debris via use of visual processing and controls in coordination with COTS hardware. The KESSLER visual processing design consists of functions developed in the MATLAB environment to process images and 3D point cloud data to provide a singular location and orientation of a desired grappling feature. This is done using a predefined feature database and various matching methods to provide the location that the end effector of the physical hardware is to be navigated to. The Controls system has utilized the ROS platform in order to design a flow of information processing to take that single point and orientation data provided by visual processing and plan a trajectory for the physical hardware to execute. In order to verify the KESSLER system success, much ground support and external resources must be employed. KESSLER location and execution accuracy relies on the Vicon lab for that verification and validation. Once all tasks are completed, the KESSLER system will be capable of performing to the standards and requirements stated in table 1. KESSLER can then be utilized to improve the LEO environment.

Acknowledgments

The KESSLER team would like to thank: Josh Stamps and Sierra Nevada Corporation (project customer), Dr. Jade Morton (project advisor), Taylor Way (financial lead team member), the Ann and H.J. Smead Aerospace Engineering Sciences department, the ASEN 4018-4028 Senior Design 2018 Project Advisory Board, the University of Colorado Boulder Vicon lab (high accuracy verification testing support) and Pacific Bell's Inc.(project financial support).

References

[1] David, L., "Ugly Truth of Space Junk: Orbital Debris Problem to Triple by 2030," , 2017. URL https://www.space.com.

[2] Center, W. D., "Kinect for Windows," , 2017. URL https://developer.microsoft.com/en-us/windows/kinect.

[3] S, O. A., "Orbbec Astra, Astra S and Astra Pro," , 2017. URL https://orbbec3d.com/product-astra.

[4] Zone, I. S.-D., "Intel RealSense Camera SR300," , 2017. URL https://software.intel.com/en-us/realsense/sr300.

[5] Mathworks, "Image Processing Toolbox," , 2017. URL https://www.mathworks.com/products/image.html.

[6] CrustCrawler, "Crustcrawler Pro-Series Robotic Arm Components," , 2017. URL http://www.crustcrawler.com/products/ProRoboticArm/.

[7] Robitis, "Dynamixle," , 2017. URL http://www.robotis.us/dynamixel/.

[8] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A., "ROS: an open-source Robot Operating System," *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, 2009.