COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

Team Jazzy

Judith Rodriguez
Yamato Matsumura
Tucker Grubbs
Alan Wang

Revised June 16, 2026

CSCI 370 Summer 2026

Prof. Bodeau

Table 1: Revision history

Revision	Date	Comments
New	5/18	Completed Sections: - Introduction - Functional Requirements - Non-Functional Requirements - Risks - Definition of Done
Rev – 2	5/27	Completed Sections: - Team Profile - System Architecture Edited: - Changed bullet points to numbers - Added Risk Register
Rev – 3	5/29	Edited: - Risk Register - Technical Design Issues
Rev - 4	6/01	Edited: - Risk Register introduction Completed Section - Software Test and Quality section
Rev - 5	6/02	Completed Section: - Software Test and Quality
Rev - 6	6/12	Edited: - Software Test and Quality Added Section: - Project Completion Status - Future Work - Lesson Learned
Rev - 7	6/16	Edited: - All current sections (present -> past tense) - System Architecture - Software Test and Quality - Project Completion Status - Future Work - Lessons Learned Added Section: - Acknowledgements

Table of Contents

I. Introduction

Stratom specializes in various autonomous applications such as developing unmanned ground vehicles, robotic systems, robotic refueling, and cargo movement. They aim to help their clients optimize uptime and make processes more efficient through automation. Through these goals, they are able to address labor shortages, increase profitability, and help keep people safe.

The team's project focused on the cargo movement application of Stratom. The project's main goal was to recreate scenes captured via a camera in a 3D space. Furthermore, the mentioned recreation was intended to be an accurate representation of the scene through the use of filtering and hole filling. This program would allow autonomous robots to get a better sense of their surroundings as well as be able to more accurately and precisely interact with cargo.

II. Functional Requirements

The team's project had three main components: Capturing the scene, generating a point cloud, and reconstructing the scene on 3D software. To note, a point cloud is a 3D point representation of the originally captured scene. This allows the pipeline to recreate captured scenes into a clean 3D model for robotic simulation.

1. Capturing the scene
    a. To capture the scene, a depth map with normals must be defined to ensure the scene is accurately represented.
2. Generating a Point Cloud
    a. After generating a point cloud, the points are used to fully define the object. Specifically, the point cloud is pre-processed by filtering out outliers and noise. The resulting product would be a cleaner, filtered point cloud of the object.
    b. Once this was acquired, meshing was used to smooth out the point cloud and define boundaries for our object in 3D.
    c. Following this, the post-processing step would fill holes in the object and ensure the normals on the object are consistent.
3. Reconstructing the 3D scene
    a. Using the generated mesh, Gazebo would be used to simulate the object to ensure the resulting 3D scene is accurate to the original photo.

III. Non-Functional Requirements

Some non-functional requirements that are outlined for the project include triangle count, software integration, language considerations, and reliability/stability.

1. Triangle Count
    a. During the creation of the mesh, the team would ensure that there are enough triangles to accurately represent the object. This is because too many triangles could result in slow run times whereas too few results in the loss of important details.
2. Software Integration
    a. The project used Docker and ROS2 to maintain the industry standard and software used by Stratom.
        i. Docker is a software that helps to simulate different operating environments. This helps build, manage, and distribute our software.
        ii. ROS2 is the robotic industry standard middleware for sensors to communicate to robotic components through software programs
3. Programming language consideration
    a. C++ and Python were both under consideration.
        i. C++ is more efficient overall, with faster runtime and processing.
        ii. Python is better for prototyping and testing, especially with a new camera
    b. In the end, the team went with Python as it was easier to use and allowed for rapid prototyping to iterate on ideas.
4. Reliability and Stability
    a. An accurate mesh and simulation are necessary so that the robot arm does not damage goods when the program is used.

IV. Risks

The project also had associated risks with it as well. The table below outlines those risks, including treatment plans to mitigate those risks.

Asset	Vulnerability	Threat	Likelyhood	Impact	Risk	Treatment	Control	Mitigated Risk
Camera	Damage	Water, outside factors	Unlikely	Doomsday	5	Reduce Likelihood	Keep camera safe and secure at all times and away from water	2
Camera cord	Damage	Extension, water	Unlikely	Medium	2	Reduce Likelihood	Keep coord away from water	1
Docker container	Lose current instance	Poor code management	Likely	Serious	12	Avoidance	Use branches when doing major/unsure changes	5
Python nodes	Losing code, etc	Non-saving, deletion, etc	Low	Serious	6	Reduce Impact	GitHub will keep working versions so less will be lossed when pushing	1
Camera images	Unauthorized access to pictures	Library security leak	Unlikely	Doomsday	5	Reduce Likelihood	Cybersecurity precautions (Talk to Stratom)	1
Image quality	Not enough points	Remove necessary data points used in pre processing / not enough data	Medium	Medium	6	Reduce Likelihood	Constanty be looking at the details of the picture	1
Team quality	Communication Breakdown	Full takeovers or not putting in work. Client stops responding or updates stop	Low	Serious	6	Avoidance	Make sure everyone is pitching in and that we continue to email updates to Stratom and our advisor	3
Target object	Damage	Inaccurate mesh recreation	Medium	Catastrophic	12	Reduce Likelihood	Make sure to test extensively to make sure meshes are accurate	5
Code	Node breaks down and stops communicating with other nodes	Bugging code/edge case and the arm fails	Low	Serious	6	Reduce Likelihood	Test edge cases and make sure code runs smoothly	3
Images	Too many points	Too many triangles in a mesh recreation may slow down the program	Medium	Medium	6	Avoidance	Reduce triangle count or add a way to tune the amount of triangles	1
Docker container	Container is not closed properly	Lack of caution	Likely	Catastrophic	16	Avoidance	Only execute known commands.	3

(Table 1 – Risk Register)

The table below outlines a couple of project and product risks that are associated with the project. These provided a more general view of the risks compared to Table 1 as well as outlined some risks associated with the end product.

Project risks	Product risks
1. This is the first time that Stratom has used this specific type of camera extensively, so the reliability and functionality of the camera are not proven yet. 2. The duration of field session is only 5 weeks, so there are some details that may be overlooked.	1. The accuracy of different objects may vary significantly based on the details of each object. 2. Some unexpected events, such as weather, might hinder the quality of images taken. 3. Precise details might be missed on certain objects.

(Table 2 – Project Risks and Produce Risks)

V. Definition of Done

The definition of done for the functional requirements are as follows:

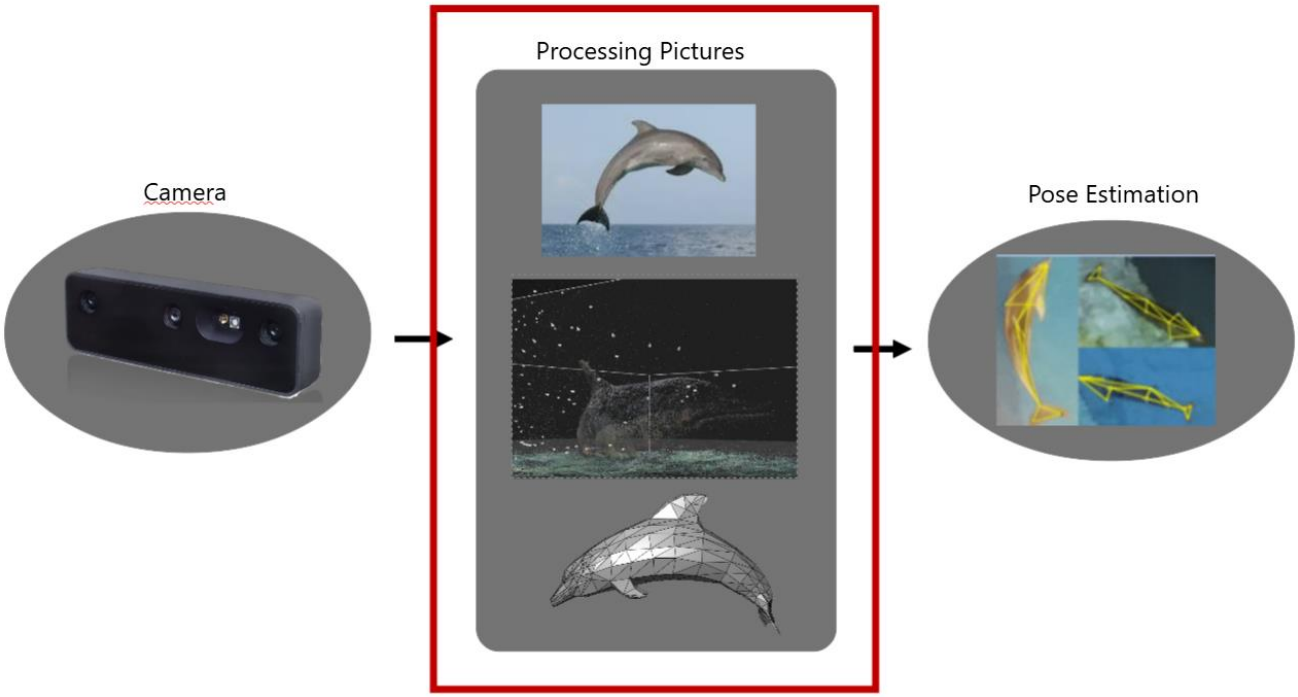
1. Capturing the Scene: A clear image of the scene along with depth data and normals
2. Generating a Point Cloud: A clean mesh representation of the object in 3D
3. Reconstructing the 3D scene: The pipeline produced an accurate model that correctly represents different simulated scenes.

Stratom will test the pipeline with images of vehicles. Thus, another definition of done would be that the pipeline is able to produce an accurate representation of the vehicle in 3D.

The product should be delivered in a Docker container containing the ROS2 pipeline for processing images.

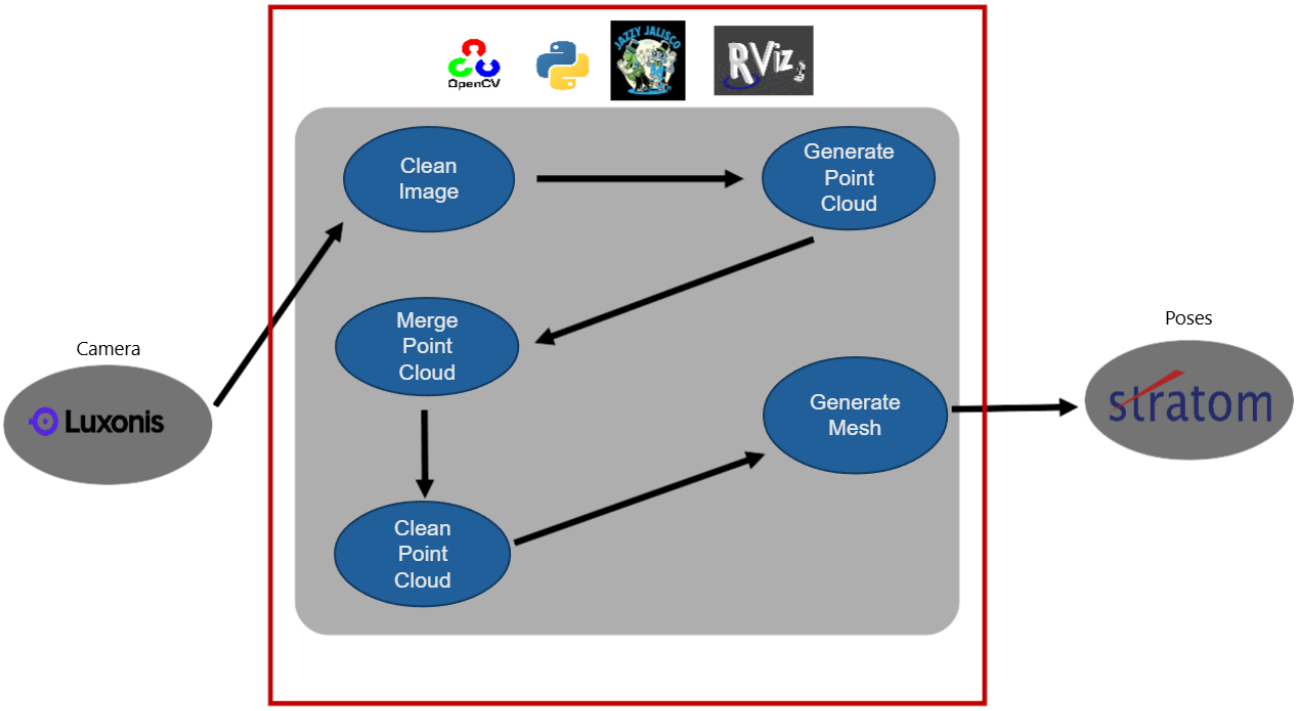
VI. System Architecture

The final product of the project was a pipeline created in ROS2. Thus, the system architecture is best represented in a linear manner. The first design below shows the system architecture at a simplified level.



(Figure 1 – Simplified Project Workflow)

Figure 1 shows the three main stages for the project. First, our camera took images of the scene. These images were then processed using a series of computer vision techniques. Finally, the processed images were used to produce a mesh of the object, and eventually, a set of poses for the robotic arm would be generated.



(Figure 2 – Image Processing Workflow)

Figure 2 presents a more in-depth figure on how the images are processed to output the robotic pose estimations. Similar to figure 1, the pipeline starts with images from our Oak-D camera. Figure 2, however, breaks down the pre-processing step into more detail. The pre-processing step first took the images from the camera scene and cleaned out the unnecessary noise. The noise factors included background, shadows, and other environmental details. The cleaned images were then sent to a node that took the cleaned images and generated a point cloud. These point clouds were then sent to another node that merged all the point clouds together. This merged point cloud was then processed to filter out duplicate and outlier points. Additionally, some of the points were cut out to reduce the processing speed when creating the mesh. Finally, the edited point cloud was used in the reconstruction of the object as a 3D mesh that will be used to send pose estimations to a robotic arm.

VII. Software Test and Quality

To ensure the quality of the product, the team produced tests for the product. Below are the functional and non-functional requirements revisited with test cases to ensure all requirements are met.

Functional Requirements Tests:

1. Capturing the scene tests
    a. Testing:
        i. To test the process of capturing the scene accurately, the team used a Linux machine to record a bag of data using ROS2 middleware. After the bag was recorded, it was sent to computers running alternative operating systems to ensure the data can be accessed and used from different machines.
    b. Pass Fail Criterion:
        i. The pass-fail criterions were mainly dependent on whether or not the bags of data were able to be distributed properly. This included the data bag containing all the needed information as well as working properly on different machines.
    c. Results:
        i. The test to send the data to other operating machines and to use the data on the other operating machines was successfully passed.
2. Generating a Point Cloud tests
    a. Testing:
        i. The generation of the point cloud was tested by ensuring that the resulting point cloud is only of the object of interest. Additionally, images with lots of background noise were tested to ensure outliers do not exist in the final 3D point cloud.
    b. Pass Fail Criterion:
        i. The criterion to pass this test was to have point clouds with the object of interest isolated as well as having no outliers in the cloud. Although the exact definition of outliers and isolating the object was up to interpretation, this test's fail criterion was having messy point clouds where no objects can be made out.
    c. Results:
        i. The test to generate a point cloud of only the object of interest was successful. We adjusted depth to ignore further away objects so that we only recorded the object of interest.
3. Reconstructing the 3D scene tests
    a. Testing:
        i. To test that the reconstructed scene was accurate, the team ensured that protrusions greater than six inches were preserved in the final reconstruction. A visual check was also conducted to ensure that all normals were pointing in the correct direction and that the object was accurately bounded.
    b. Pass Fail Criterion:
        i. To pass this test, the metric of preserving protrusions greater than six inches must be met. Additionally, the visual check mentioned above must also pass. Although the

definition of a “good” visual was subjective and vague, this test fails if the resulting 3D mesh does not resemble the initial object.

c. Results:
    i. The test failed as the team struggled with getting the mesh to display with full accuracy.

Non-function requirement tests:

1. Fast Processing Speed
    a. Testing:
        i. The overall processing time of the program needed to be fast. If the program took too much time, the system would become less useful. To test the speed of the system, each individual frame process was timed, as well as the total mesh creation time. Ideally, the frame can be processed in a second, and the entirety of the mesh can be created within a maximum of 30 seconds. To lower the processing time, one consideration would be to decrease the number of triangles that were used during the creation of the mesh.
    b. Pass Fail Criterions:
        i. This test passed if the time to create a single mesh was less than 30 seconds. Anything more than this would cause this test to fail.
    c. Results:
        i. The speed test passed with point clouds running at around the same speed as the camera. Meshing took about 5 to 10 seconds to process, which is also within our allowable times.
2. Reliability and stability
    a. Testing:
        i. To test the reliability of the model, the team used known models with known dimensions and volumes so that the generated mesh could be compared quantitatively. Because the dimensions were known, extensive testing could be done to ensure the accuracy of the model.
    b. Pass Fail Criterions:
        i. To pass this test, the resulting model must be accurate within six inches of the known object. Although the volume was not as important as dimensions, this should be accurate to roughly 90% of the original objects' volume.
    c. Results:
        i. The test failed as the team did not have enough time to extract enough precise data from the resulting mesh to see if the above criteria were met.

General Tests:

1. User acceptance testing
    a. Testing:
        i. To ensure the project passed user acceptance testing, the team tested the accuracy of the same mesh generated in different environments and lighting conditions that Stratom may encounter. Specifically, edge cases involved situations with sun, glare, and

shadows. The test was accepted if the resulting mesh from the data still met previously set performance standards under the edge cases listed.

b. Pass Fail Criterions:
    i. To pass this test, the team tested the same object mesh in different environments and observed if they looked similar. Specifically, the mesh was tested outside in the shade and in direct sunlight. This test failed if the resulting mesh did not pass the previous mesh metrics laid out above in different environmental conditions.
c. Results:
    i. The test for scanning the same object in different environments passed for the point clouds and failed for the meshes. The point cloud seemed to get the vast majority of the object aspects in sunny vs indoor conditions while the mesh struggled to maintain accuracy and consistency in different environments.

2. Code review
    a. Testing:
        i. The purpose of the code review was to ensure clean and efficient code of each node created by each team member. The review consisted of checking the functionality of each node and checking that similar conventions were followed by each team member.
    b. Pass Fail Criterions:
        i. To pass this test, all nodes created by all members of the team must work as intended and be easily readable. Custom parameters should be commented with their use, as well as if they were different from the default.
    c. Results:
        i. The test for code review ensured that code ran correctly, and it was clean and readable with code not needing to be explained due to commenting practices within coding and parameter files.

VIII. Project Ethical Considerations

The principles within ACM and IEEE that were pertinent to the development of the software involved principles 1.5, 2.9, and 3.5 from ACM. Principle 1.5 states to respect the work required to produce new ideas, inventions, creative work, and computing artifacts. The developed software required the use of open-source software. Thus, the team needed to credit the work done by the teams or individuals that developed the used open-source software. Principle 2.9 involves designing software that is robust and usably secure. Because we worked with computer software, it is possible for breeches of data to occur. For the project, this meant the program should run functions as intended and be secured to prevent intentional misuse of the software. Principle 3.5 states members of the organization should grow as professionals. This principle allows teams and individuals to grow professionally through experience. Under the guidance of Stratom, the team was able to learn about the robotics industry and the software used.

The IEEE principles that were reflected the most in the software developed are principles I.2 and I.6. Principle I.2 seeks to improve the understanding of current and newer technologies. The project involved using current open-source software with the use of newer hardware. It was important that the team understood how open-source software worked in conjunction with the hardware. Principle I.6 involves the maintenance and improvement of technical competence and only taking on technical tasks with the approval of qualified training/experience. It was imperative that the software was developed using ROS2 before being tested on the hardware. Once the team improved their skill at ROS2 and computer vision, only then was the software to be executed.

The principles within the IEEE Code of Ethics and ACM Code of Conduct that were most at risk of being violated during the development of this project relate to safety, harm prevention, privacy, and software quality. The IEEE Code of Ethics principle 1, and the ACM Code of Ethics principle 1.3 and 1.6 state that engineers should prioritize protecting the privacy of the public. The robotic system relied on camera data to reconstruct environments. If the robotic arm was to be deployed in restricted areas, the cameras may capture sensitive data or information that individuals have not consented to record.

Principle 9 in the IEEE code of ethics and principle 1.2 in the ACM code of conduct state that engineers should avoid injuring others, their property, or their reputation by malicious actions, rumors or any other verbal or physical abuses. If the scene was not reconstructed accurately, the robot could misinterpret its environment and move into unsafe areas, potentially causing damage to equipment or injury to nearby people. Additionally, if the 3D reconstruction was inaccurately built, it could have led to errors calculating the object or terrain causing potential safety hazards in the pose estimation part of the robotic system.

Principle 2.1 in the ACM code of conduct states that engineers should strive to achieve high quality in both the processes and products of professional work. The quality of the reconstructed mesh could be compromised if too many points were removed from the point cloud to improve processing speed, which could have led to important environmental details being lost and reducing the accuracy and reliability of the final reconstructed scene.

There are a few ethical considerations that needed to be addressed if the software quality plan was not implemented correctly. It was important to think about what could go wrong in the workplace. For instance,

what would happen if the software did not account for hardware damage that could affect the movement of the robotic arm? How could this affect the people around the arm? How could this potentially damage cargo being moved within the arm? An example of why it was important to address ethical considerations can be seen when a worker was hit and injured by a robotic arm at Tesla. The technician had to get medical attention and ended up suing Tesla and the robot maker [1]. For these reasons, ethical considerations are important to the discussion of software quality, testing, and safety.

There was one major security consideration in the project. The product used publicly available, open-source Python libraries to take the image data provided and turn it into a 3D point cloud and eventually a mesh. If these Python libraries had backdoors to take and send data to the creator of the libraries, the images could potentially be leaked. For instance, companies with military applications would not want the images taken by the product of their bases, vehicles, or cargo available to the public.

To ensure the project was ethical, the team followed tests outlined by Michael Davis. They outlined numerous tests to ensure engineering projects take into account their ethical considerations. Thus, for this project, the team applied two of the Michael Davis tests to ensure the project is ethical.

1. Harm Test - The team applied the harm test by examining if the potential harm of having an autonomous robotic arm maneuver cargo outweighs the benefits. The team identified that a potential harm of the project would be the robotic arm accidentally damaging cargo or hitting someone nearby. However, a big benefit of the project was that the cargo loading process would become more efficient by eliminating the human error component as well as reducing labor requirements. Ultimately, the team determined that these benefits were worth the potential risks involved and thus passed the harm test.
2. Common Practice Test – The team applied the common practice test by asking if everyone automated the cargo loading process, would it be beneficial? It was determined that this would lead to more efficient and precise cargo loading. Additionally, this would allow an increase in uptime for the process, allowing faster transportation of cargo as a whole. Thus, it was concluded that the common practice test was passed.

IX. Project Completion Status

The goal of Team Jazzy's project was to create a clean mesh using 3D reconstruction processes so they can be streamlined to robotics applications. The team approached this through the use of two open-source software programs: RTABMAP and Industrial Reconstruction.

Overall, the team was able to accomplish many of the goals for the project. For example, we were able to properly use the camera to record bags of data. Additionally, both RTABMAP and Industrial reconstruction were able to mesh using live as well as pre-recorded data from the camera. The team learned a lot about configuring hardware to communicate effectively with the software.

However, the team also fell short on a couple of goals for the project. First, there were problems with post-processing. Although the software was able to recreate meshes, they were not clean and oftentimes were hard to visualize. Thus, one feature that the team did not have time to implement was having live, accurate 3D meshes. Additionally, the final step of pose estimation was unable to be accomplished. In terms of testing results, the initial tests with data bags and point clouds passed. However, the inability to create an accurate 3D mesh caused the last few mesh quality tests to fail. Additionally, the team did not have enough time to test in different environments as the team was unable to create an accurate mesh in a controlled, indoor environment. Usability tests also failed because the team did not have a fully working pipeline for Stratom to implement into their use cases.

X. Future Work

Further work that can be done on the project include getting increased accuracy for the mesh, ensuring the software properly functions in different environments, implementation with Gazebo, and pose estimation. These could be implemented in numerous ways. Greater mesh accuracy would come from further parameter tuning within the RTABMAP and Industrial Reconstruction libraries. Use of the IR laser that the Oak-D Pro is equipped with should help keep mesh accuracy stable in different environments. Furthermore, if an accurate mesh can be created, pose estimation would follow naturally as the chance for inaccurate poses is lowered from having an accurate mesh. Gazebo is a robotics simulation software. The generated mesh would be used within Gazebo to test external factors such as rain, snow, and low light conditions through simulation.

XI. Lessons Learned

There were many lessons learned during the process of creating the product. The first and biggest lesson was to read documentation extensively before rushing into the coding process. There were a few times where the team got stuck and had to stop to fix a bug. When this happened, the team was quick to try fast fixes until we exhausted our options over the course of one to two hours, before switching to reading documentation. Once the documentation was read, the team quickly found the solution.

The second biggest lesson learned was that hardware to software communication is difficult and time consuming. The team was not given access to the Oak-D-Pro until week two, and it was assumed that it would take a day at most to setup. The camera was unable to communicate with the computers/software until week three when the team switched to running Linux. This process took up much of the limited time and ultimately contributed to the incomplete results of the project.

Some of the other lessons that were learned are as follows:

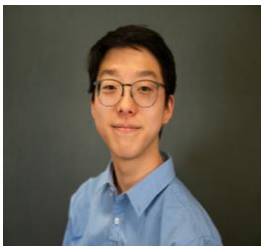
- Guidance is not always available. There might be times when there is minimal documentation or active help. It was tough in these situations to not feel bad about what felt like very little progress.
- Team communication is very important. The team would often stop to talk about different pieces of code. After going through this process multiple times, many bugs were avoided by talking through code and parameters.
- The terminal is super helpful when debugging. Most of the time it told us where and what the specific issues were. Even when the terminal did not tell us exactly what was going wrong, there were typically very big hints as to what the problem was so that a fix could be made.

XII. Acknowledgments

We would like to thank our client, Stratom, for providing us the opportunity to work on such an interesting project. We would like to specifically thank Kristin Farris for all of the support throughout the project. Her guidance was exceptionally helpful to navigate the challenges we encountered along the way.

Additionally, we wanted to thank our advisor, Donna Bodeau, for all of the encouragement and clarity to help us understand the project and to keep us on track. Her mentorship was invaluable and helped us grow professionally throughout the course.

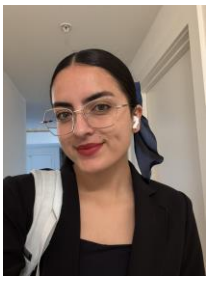
XIII. Team Profile



Yamato Matsumura
Junior - Colorado School of Mines
Computer Science - Data Science
Hometown: Oxford, Mississippi
Work Experience: Vermeer Data Analytics Intern, IT Service Desk Student Manager
Interests: Piano, Volleyball, Racquetball

Alan Wang
Junior - Colorado School of Mines
Computer Science – Space
Hometown: Highlands Ranch, Colorado
Work Experience: CSM Lunabotics Team
Interests: Riftbound, Magic the Gathering, Gaming

Tucker Grubbs
Junior - Colorado School of Mines
Computer Science - Robotics and Intelligent Systems
Hometown: Fort Worth, Texas
Work Experience: Swim Instructor, Petsitter
Interests: Volleyball, Game development, Games with friends



Judith Rodriguez
Junior - Colorado School of Mines
Computer Science - Robotics and Intelligent Systems
Hometown: Chihuahua, Mexico
Work Experience: Sales Rep, Math Tutor, CS Tutor
Interests: Volleyball, Baking, Pottery, Poetry

References

[1] N. Owens, “Former factory worker sues Tesla after robotic arm knocks him unconscious,” ManufacturingDive, Sept. 25, 2025. Accessed: June 2, 2026. [Online]. Available: https://www.manufacturingdive.com/news/former-factory-worker-sues-tesla-fanuc-robotic-arm-unconscious-51-million/761123/

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations and acronyms

Term	Definition
Point Cloud	*A collection of 3D data points. For the project, these are data points captured by the camera for the scene.*
ROS2	*Robotic Operating System 2*
Mesh	*A 3D geometric structure representing the surface of the object. For the project, these are done to turn the point clouds into a 3D surface.*