University of Arkansas – CSCE Department
Capstone I – Final Proposal – Fall 2022

NASA Robot Mining Competition

Ahmed Moustafa, Rohit Kala, Justin Kilgo, Jackson Newman, and Jackson Burger

Abstract

At the current growth in population, we will soon exceed the planet's non-renewable resource availability. This vast difference between demand and supply necessitates a new source for scarce minerals: space. For these extraterrestrial mining expeditions, mankind must apply advanced technology to develop lunar robots to kickstart the era of space mining. The objective of this project is to engineer and program a robotic lunar excavator for the NASA Lunabotics competition that will autonomously maneuver through complex, rough terrain and mine regolith simulants. As the CSCE section of the NASA Robot Mining Competition Capstone team, our focus is on the software and computer system aspects of the project through simulation and physical testing to achieve autonomous function. To successfully accomplish this task, we plan to use both C++ and Python with ROS2 (Robot Operating System 2), which is an open-source library that will help interface with the low-level hardware of the robot. A key aspect of the autonomy of the robot is position-tracking which will use a combination of accelerometer measurements and camera object detection. With a dataset of possible field elements and regoliths, we will train a model using modern object detection libraries to facilitate autonomous navigation so the robot can safely mine. This project's significance lies in both the direct cross-department learning opportunities as well as the general impact this field of technology will have on our planet's future. Working in unison with the University of Arkansas' Razorbotz team highlights the significance of multifaceted engineering projects and the breadth of knowledge required to reach an acceptable product. The robot and autonomous programming we plan to produce are the foundation for future lunar and martian mining expeditions to help alleviate Earth's resource exhaustion and environmental damage from current resource extraction.

1.0 Problem

At the rate at which Earth's population is consuming non-renewable resources, mankind will not be able to sustain its growing population [20]. One of the options mankind has been looking into is searching for resources off-planet. A prime candidate due to its proximity and signs of resources such as water and Helium-3 is the moon which led NASA to create the Artemis program [21]. NASA's Artemis program plans to establish the first human presence on the moon since the 1972 Apollo 17 mission. These plans include the development of a lunar base camp which would possibly be built in Shackleton Crater due to its suspected access to ice and minerals. There are many concerns with the Artemis missions, especially surrounding the sustainable deployment of humans to the moon. It is both dangerous and expensive to send people to the moon with each pound of commercial cargo costing anywhere from $9,100 to $43,180 [22]. This is one of the main reasons behind the development of and demand for precursor lunar robots. To facilitate human travel to the moon, Mars, and other extra-terrestrial sites, we need to engineer efficient robots that can accomplish the pre-requisite task of mining and research. This is where the problem that our project, the Lunabotics competition, and the NASA Artemis program all hope to address arises.

Lunar vehicles have been in the works since the 1950s and experienced a huge increase in development in the 1960s thanks to the Space Race between the US and the USSR during the Cold War. As NASA went through many iterations of lunar robots, the expectations rose from travel and navigation to mining, on-site research, and continuous exploration. A key problem with deploying robots in space is their dependence on manual control. For a radio signal to reach the moon, it can take several seconds which does not seem like much but to reach a robot on Mars from Earth, it can take around 7 minutes [24]. Because of this time delay, human control would be difficult and possibly dangerous for the robot since any decisions/movements made from Earth would be anywhere from 7-20 minutes delayed. This is where the development of robust autonomy for lunar and martian robots becomes crucial. As such, NASA's Artemis Student Challenge was created to be a college-level competition where teams must engineer the most efficient robot that can reliably navigate and mine the simulated lunar environments autonomously.

The importance of developing an efficient autonomous robot for space mining and exploration grows with every second. This challenge is something that NASA has been trying to solve for several decades and mankind is rapidly approaching a deadline. The impact of not solving this problem of efficient precursor lunar robots is the exhaustion of resources on Earth and the growth in environmental risks of our current on-planet mining. Our current mining processes for rare-earth metals are causing contamination of air, water, and soil due to leaching chemicals and toxic waste [25]. If mankind isn't able to rapidly reduce its consumption of the resources in question and the environmental consequences of their extraction, we will only be digging our graves. To combat our growing population and resource demands, off-planet mining becomes almost necessary and so the challenge of researching and developing the most efficient and inexpensive autonomous mining robots is imperative.

2.0 Objective

The objective of this project is to engineer and program a robotic lunar excavator for the NASA Lunabotics competition that will autonomously maneuver through complex, rough terrain and mine and extract regolith simulants. Alongside the autonomous functionality will be a realistic backup manual control using joysticks and a simulated delay to match the delay between the controllers on Earth and the robot on the Moon.

3.0 Background

3.1 Key Concepts

3.1.1 Object Detection

The latest advancement in artificial intelligence has allowed us to solve previously difficult problems. One such task is object detection, which can be described as the task of detecting individual instances of various objects found within an image or a video and classifying them correctly [14]. It is one of the most pertinent challenges in computer vision, as many problems in the field require object detection to accomplish various other tasks such as pose estimation, image segmentation, etc. There are a lot of factors that have allowed us to finally start solving this task. Firstly, hardware has improved to the point where we can implement multilayer perceptrons, which can be thought of as neurons represented in a computer. Multilayer perceptrons were first theorized during the 1950s [15], however, it was impossible to implement them efficiently with the technology at the time. With the rapid improvement of technology during the 21st century, we've finally achieved implementation. The reason why perceptrons are so important is that we can solve problems without having to explicitly code for them. Rather, we use data that we know the results of and allow the computer to come up with a function or a model so that it can solve this problem on its own. This is because a perceptron will fire if the inputs scaled by a weight manage to exceed a threshold amount [16]. The computer can solve problems based on the perceptrons that fire, thereby creating solutions to a new set of problems that were previously hard or impossible to solve. A model might take many perceptrons to solve harder tasks, and they are organized into layers, where each perceptron takes input from the previous layer [16]. A crucial part of using such models is to have a dataset that it can run over so that the model can optimize the amount by which the inputs are scaled such that the model outputs the correct result. Each incorrect answer will further optimize the model by shifting the scaling factor by a small amount, eventually converging at an optimal value so that the model is accurate for the training data [16]. It is also important to test the model on randomized data to ensure that the model can accurately predict the output to various kinds of input rather than overfitting to the training dataset [17].

While multilayer perceptrons are very good at solving rudimentary tasks, we will need something more complex to solve tasks such as object detection efficiently and accurately. This is where the idea of a Convolutional Neural Network comes in. Essentially, we will have certain layers in the network which will convolve a kernel matrix with portions of the input data. These layers will extract the various features of the input, therefore, allowing the model to classify these features. In these layers, the kernel matrix is optimized so that the model extracts the features correctly [18]. This method is commonly used in object detection and implementations often try to maximize speed or precision. Oftentimes, these are inversely related as faster models tend to be less precise. A primary drawback of these models is that they require lots of data to predict the answer accurately. Additionally, training some of the more complex models takes a long time and requires lots of calculations. One of the best datasets for the object detection task is Microsoft's COCO dataset, which is a joint project by many companies and universities to classify various objects in everyday scenes that anyone can use to train a model [19].

3.1.2 Client-Server Communication

Client-server communication is commonly used when one system cannot access the resources of another system. It allows a system to share necessary information with another so that each system does not have to store duplicate information. In this relationship, there is a client and a server as the name implies. The client is primarily responsible for requesting and receiving information, while the server is responsible for handling this request and responding with the appropriate information. One way this is accomplished is through the Transmission Control Protocol (TCP). This allows the client and server to first establish a connection before transmitting data to ensure the information requested is sent and received to the correct destination. First, the client sends an initial request to begin this connection. Second, the server acknowledges the request so that the client knows a connection is being established. Next, the client sends an acknowledgment of the connection so that the server knows to begin transmitting the data. TCP connections may persist for future communication, however, they are commonly terminated due to the number of requests many servers receive. In the case where the connection persists, information can continue to flow from the server to the client with no interruptions. In the case where the connection is terminated, the connection must be reestablished later before more data may be transmitted. The application of this Client-Server model is useful in many multi-system technical methods such as transmitting control signals for motors or publishing images from a camera.

3.2 Related Work

3.2.1 Servi Robots

Service robots are becoming increasingly more common with the advancement of artificial intelligence. The robot which we are programming, at a high level, should autonomously drive itself to a location while avoiding obstacles to do a certain task, and then return to where it started. This is closely related to a recent development from Bear Robotics partnered with Softbank Robotics. They created a service robot, called Servi, which autonomously delivers food to customers in a restaurant as well as busses tables [1]. This Robot as a Service (RaaS) allows businesses to replace existing employees with a much cheaper and seemingly more reliable option, or add the robot to their staff. This is related to the robot which we are programming because Servi, at a high level, drives itself to a table, does some task whether it be to serve customers or bus the table and drives itself back to where it started all while avoiding obstacles. Although this is a good solution to the inconvenience that having a human employee can cause, it is not a perfect solution. The Servi robot is simply a moving tray. Some human intervention is still needed for it to operate how it was intended to. People still need to load and unload it, and while bussing is an ability that Bear and Softbank are advertising it can do, it can only hold the dishes, not pick them up. The robot we are programming should be able to do all of its tasks without any human intervention. When it gets to its location, it will dig up rocks and store them without needing the help of humans.

Autonomous robots will likely replace many humans in a lot of different fields of work. They have great benefits for many industries due to their productivity and precision. With a precise robot, there are next to no human errors that can occur which would decrease efficiency or set back a company. Transporting goods is the most relevant task where we can see the potential of autonomous robots. Not only can the robot be a solution for human error, but it can also result in more safety for humans. With regards to having an autonomous robot mine on worlds other than

Earth, the benefits of autonomous robots become even more apparent. Instead of having a human go into space and mine by hand, NASA can send a robot or even a group of robots to do the same thing faster without the risk of the human being harmed or becoming fatigued and less efficient. There are some obvious concerns when replacing humans with robots. There is always the possibility that the robot does not work the way it was intended to. If this should happen, then it can cost a company a potentially very large amount of money, it can slow down progress, or it can even put humans at risk if they are working around the robot if it were to fail. The solution to this is to test the robot extensively before and during production, implement safety features that can immediately stop a robot so a potential problem can be investigated, constantly search for ways of improving the robot, and study the robot's performance closely.

3.2.2 Roomba

Autonomous robots like our lunabot are nothing new. For example, Roomba by iRobot has been in many homes across America since 2002 [2]. While it is much smaller, it utilizes many technologies that will need to be considered for the lunabot, primarily calculating location and path using infrared emitters and cameras. While the Roomba is running, it is mapping out the area and uses this information to plan efficient cleaning courses and remember where the robot has been. This allows it to run for hours since it can automatically return to its home dock and recharge when the battery is low, and then return to the location and path it was on before the battery was low [4]. While the lunabot will not solely use a camera to understand its location, it is a large source of its direction and enables object detection. We will need to use the camera in a similar way to avoid rocks and potholes while still making progress toward our goal. However, we need to identify and excavate regolith whereas the Roomba is focused solely on the navigation of its field. This difference in task complexity signifies the additional challenge in the autonomous development of our robot. Also, we do not have the luxury of moving around the map and bumping into things to map out the location like the Roomba, we must be able to navigate it our first time through and avoid hitting things to prevent potential damage to the robot. Luckily, we can make use of a more powerful camera and more advanced frameworks for the robotics competition, but that does not cancel out the additional requirements that we are held to.

4.0 Approach/Design

4.1 Requirements

The competition has a strict set of rules for the robot that must be followed. These rules are the set of requirements that goes as follows ([3], page 29-30):

- The robot, before moving, must be contained within the dimensions: 1.1m length x 0.6m width x 0.6m height. After it begins, it may expand beyond these dimensions, but can not exceed 1.5m in height.
- The robot must not be any heavier than 80kg.
- There must be at least 4 points which humans can lift the robot by. These points must be clearly marked (ISO 7000-1368) so the robot can be safely moved.
- The robot can be launched in any orientation. The axes X, Y. and Z correspond to length, width, and height and must be declared to the inspection judge.

- The robot must have a Reference Point Arrow marked on it which points in the direction that the robot should begin moving in before it starts. This arrow does not indicate what direction the robot should always move in.
- Subsystems which are attached to the robot that transmit commands, data, or video will be counted toward the final weight. Equipment used for these purposes that are not attached to the robot will not be counted in the final weight.
- Multiple robots are allowed, but must comply with the dimension and weight specifications as 1 unit.
- The robot may be controlled either with a remote control or autonomously.
- Touch sensors are not allowed.
- The robot must be equipped with a “Kill Switch”. This Kill Switch must be a red, easily accessible emergency stop button which satisfies the following conditions: It must be implemented in a way that it uses trusted engineering practices and principles. It must have at least a 40mm diameter. It must be placed on a surface that is easy to access and requires no additional steps to get to. Only 1 Kill Switch may be placed on each robot. Disabling the Kill Switch without authorization from the staff will get the team disqualified. The Kill Switch must immediately disable the robot after 1 press of the button. It must be extremely reliable. Due to these reasons, a Commercial Off-The-Shelf red button is required. A closed control signal to a mechanical relay is allowed if it stays open to disable the robot. The button should disconnect the batteries from all controllers and it should isolate the batteries from the rest of the active sub-systems. If a system is powered by its own, independent, internal computer battery, then it may stay powered on in the event of the Kill Switch being pressed.
- The robot must provide its own power. No power from the facility will be able to be used for the robot during the attempt. The robot can use as much power as it wants, there is no limit. The energy used by the robot must be recorded with an Commercial Off-The-Shelf logging device. The energy used must immediately be shown to the judges after the attempt.
- The robot can not use any processes, gasses, fluids, or any consumables that would not work at a place other than Earth. Something like a dust wiper controlled from Earth that operates on the robot is acceptable. Closed pneumatic systems can be used by the robot if the gas is supplied by the robot.
- Due to limited budgets, the robot does not have to use equipment that is able to be used off-world, but only if the components can be swapped for off-world environment rated equipment. Examples of equipment that the robot cannot use are: GPS, rubber pneumatic tires, air/foam filled tires, open/closed cell foam, ultrasonic proximity sensors, or hydraulics due to the fact that NASA cannot anticipate these to be used in an off-world mission.

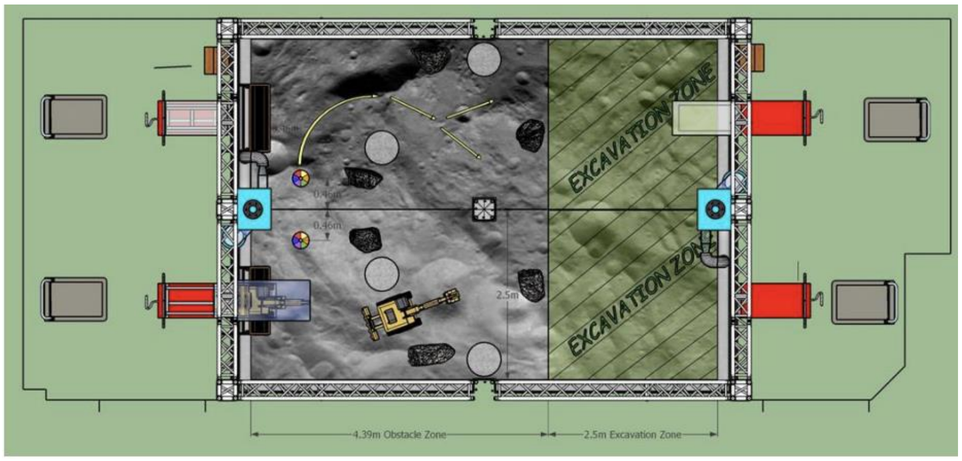

Figure 1 - Mining Arena ([3], page 32)

4.2 Use Cases

Use Case #1: Gather resources from dangerous/inaccessible terrain
Author: Jackson Newman
Primary Actor: NASA researcher/Robot supervisor
Goal in Context: To gather information based on samples that the robot retrieves or to use the resources that are found outside of Earth or in extreme environments on Earth.
Preconditions: Regardless of where the robot is mining, the robot supervisor must be well-educated and prepared to ensure the robot's mission is successful. If the robot is going into space, there must be proper extra-terrestrial transportation and the researchers must be well-equipped to conduct useful research with the data.
Trigger: There is a demand for resources from dangerous environments or off-planet. If from space, NASA may request for research to be done on certain resources or environments to prepare for human space travel.

Scenario:
1. If the mission is for space, NASA sends the robot to another planet.
2. The robot is supervised whilst autonomously moving through the terrain and gathering resources.
3. Once the robot has gathered a satisfactory amount of resources, it and its harvest return from the possibly dangerous or off-planet dig site.
4. The resources are then studied or used.

Exceptions:
1. The robot becomes obsolete because other research or resource extraction surpassed the robot's in importance, value, and/or simplicity.
2. The regions are too difficult to reach or the resources simply don't exist in feasible quantities of nearby terrain/planets.
3. The resource may not be necessary enough to justify spending the money to send the robot on a trip to collect it.

Priority: Medium
Channel to Actor: Transport Vehicle (Spaceship if extra-terrestrial, Truck/Boat if terrestrial)
Usage Frequency: Yearly
Secondary Actors: Robot wheels, robot scooper

Channels to secondary actors: Programmed directions to the dig site and amount of dig time.

Open Issues:
1. What if the robot fails while on its mission and is too far gone for retrieval?
2. What if the demand for the excavation becomes irrelevant while the robot is in transit/operation?

Use Case #2: Explore and excavate terrain to rescue trapped humans
Author: Ahmed Moustafa
Primary Actor: Rescue operator/Robot supervisor
Goal in Context: To navigate through dangerous terrain and create a safe path for humans to enter/exit for the sake of rescue.
Preconditions: The operators of the robot and rescue are ready for an emergency and there exists a terrain the robot can excavate to enable further rescue efforts. The current design/expectations of the robot will have to be slightly altered to further match this use case and support rescue operations.
Trigger: There is an emergency where people are trapped in terrain like caves or ravines that are inaccessible to normal rescue methods. The trigger may also be a precautionary one where the robot is a companion to human navigation of an environment known for trapping humans.

Scenario:
1. If there is a person in need of rescue or navigation assistance, the robot will be deployed to the location.
2. The robot is supervised whilst autonomously moving through the terrain and excavating paths for people in difficult-to-maneuver terrain like caves/mines.
3. Once the robot has cleared and navigated paths for entrance/exit the person can safely explore or be rescued as necessary.

Exceptions:
1. The terrain is too difficult for the robot’s design (space is too small for even a robot or there is too much water). It’s possible that a design geared more toward rescue would solve this.

Priority: Medium
Channel to Actor: Transport Vehicle
Usage Frequency: On Demand (perhaps monthly)
Secondary Actors: Robot wheels, robot scooper

Channels to secondary actors: Programmed to assist the explorer/trapped human.

Open Issues:

1. What if the robot fails while on its mission and we waste time on an incorrect rescue method?

4.3 High Level Architecture

The NASA Lunabotics challenge places a heavy emphasis on the autonomy of the robot. Therefore, it shall also be our primary focus to ensure that the robot is autonomous to maximize the team's points in the competition. The preliminary task to ensure that the robot is autonomous is to ensure that it can understand its spatial location as it traverses the competition ground. In other words, the robot must be able to navigate the course by itself. To accomplish this task, we will have many sensors that measure various movement metrics of the robot such as the SparkFun LIS2DH, which will measure the acceleration of the robot without consuming much power. It contains capacitive plates, which move in relation to each other as the system accelerates. This changes the capacitance, which is proportional to the acceleration of the system, thereby, giving the robot a way to measure its acceleration [5]. It is a 3-axis accelerometer, thus, it can measure the acceleration in the x, y, and z planes of motion [6]. The robot's movement will be powered by various motors. Additionally, we will have encoders gather other metrics such as the speed and direction of the robot. Next, we will use a ZED2i stereo camera to ensure that the robot can avoid various obstacles that will be present throughout the competition's course. The first advantage of the ZED camera is that it can estimate the depth of various objects using binocular front-facing cameras, much like how human eyes function (illustrated in Figure 2). Another advantage of the camera is that it can map the entire 3D plane that the camera sees, thus giving our robot a way to create a map of the entire competition ground so that it can optimize its traversal by finding the path of least resistance from the extraction zone of the regolith to the target hopper. The camera augments the estimated location of an object from the front-facing cameras using a neural network [8]. The camera will also feed all the information to the team's controlling computer, thus we can see what the robot sees, in addition to the robot's depth map. This gives us a way to manually control the robot in case it runs into a situation it cannot handle by itself. However, this information is delayed by 5 seconds to simulate the vast distances the information travels during various missions that NASA undertakes, further incentivizing us to make the robot as autonomous as possible.

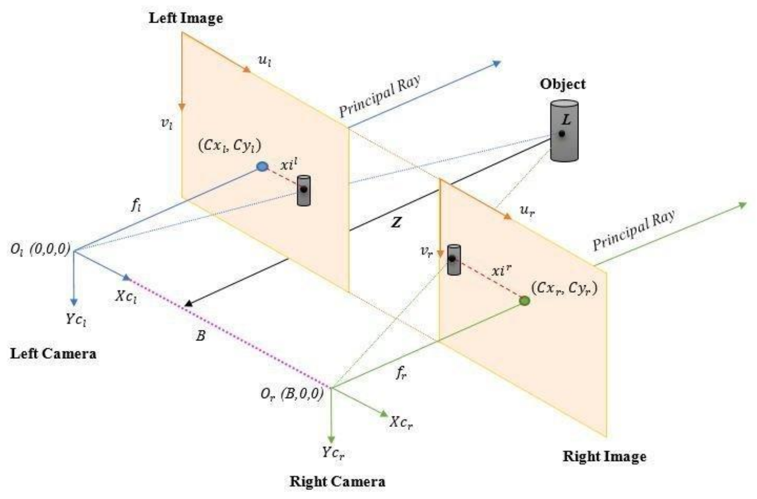

Figure 2 - An illustration of binocular vision [7]

In addition to course traversal, our robot must be able to identify key objects, i.e., certain simulated regoliths that will be found throughout the competition's course that the robot must excavate. To accomplish this, we couple the video frames taken in with the ZED2i camera with a state-of-the-art neural network designed for object detection. One of the best ways to implement a neural network is by using Python's Pytorch and Torchvision libraries [9]. These libraries contain various Python modules to easily set up and train a computer vision model for the object detection task. The team will gather pictures of these key objects and train a pre-existing model and then evaluate the performance of the model. One of the key candidates for this task is the YOLOv3 object detection model, which is a lightweight and accurate object detection model [10]. This model uses various convolutional layers to extract features from the image. To do this, YOLO breaks the frame into multiple smaller sections, then predicts a bounding box for each object, as well as the probability for the object contained within the bounding box to be of a certain class (an example is illustrated in Figure 3). In our case, each class will correspond to a type of regolith. To put it all together, the robot is able to map the entire course. Furthermore, it is able to find important regolith to extract on its own using the YOLO model that we train. Now, all that is left is for us to implement a path-finding algorithm based on the map that the robot generates so that it can extract the regolith and drop it off at the target hooper. The best choice for pathfinding in our case will be Dijkstra's algorithm [12], which will allow the robot to find the shortest path between two or more points.

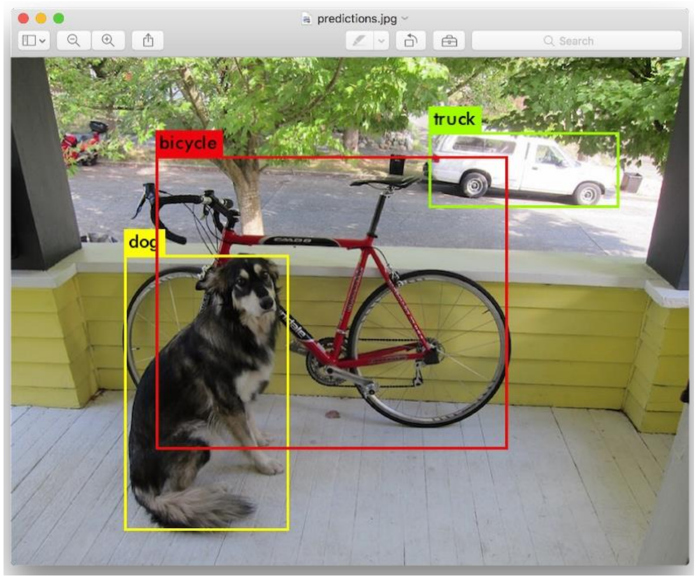

Figure 3 - An illustration of object detection task [11]

To implement all that we have so far, we will use ROS2 (Robot Operating System 2), which is an open-source library that will help interface with the low-level hardware of the robot. We will organize each of the aforementioned hardware into ROS Nodes which is a fundamental element that serves a single, modular purpose. This will allow each hardware component to send and receive information between each other either through topics or services and use this information to power motors or do tasks. Before we go into more detail, we will describe the difference between a topic and a service in ROS2. A topic can be viewed as a publisher/subscriber model, where each node who is subscribed to a topic will gain updates on new information from each publisher of that particular topic. Comparatively, a service can be viewed as a server/client model between nodes, where a client node requests service from a server node. The primary difference between them is that the updates on information are continuous between topics, whereas the information is only granted upon a request from a client node in a service[13]. With this in mind, we will organize the robot into the following abstract nodes (illustrated in Figure 4): Remote Client Communication (RCC), Joystick Commands to Motor Speeds, Motors, and Automation. We further break down these nodes and their connections in the coming paragraphs.

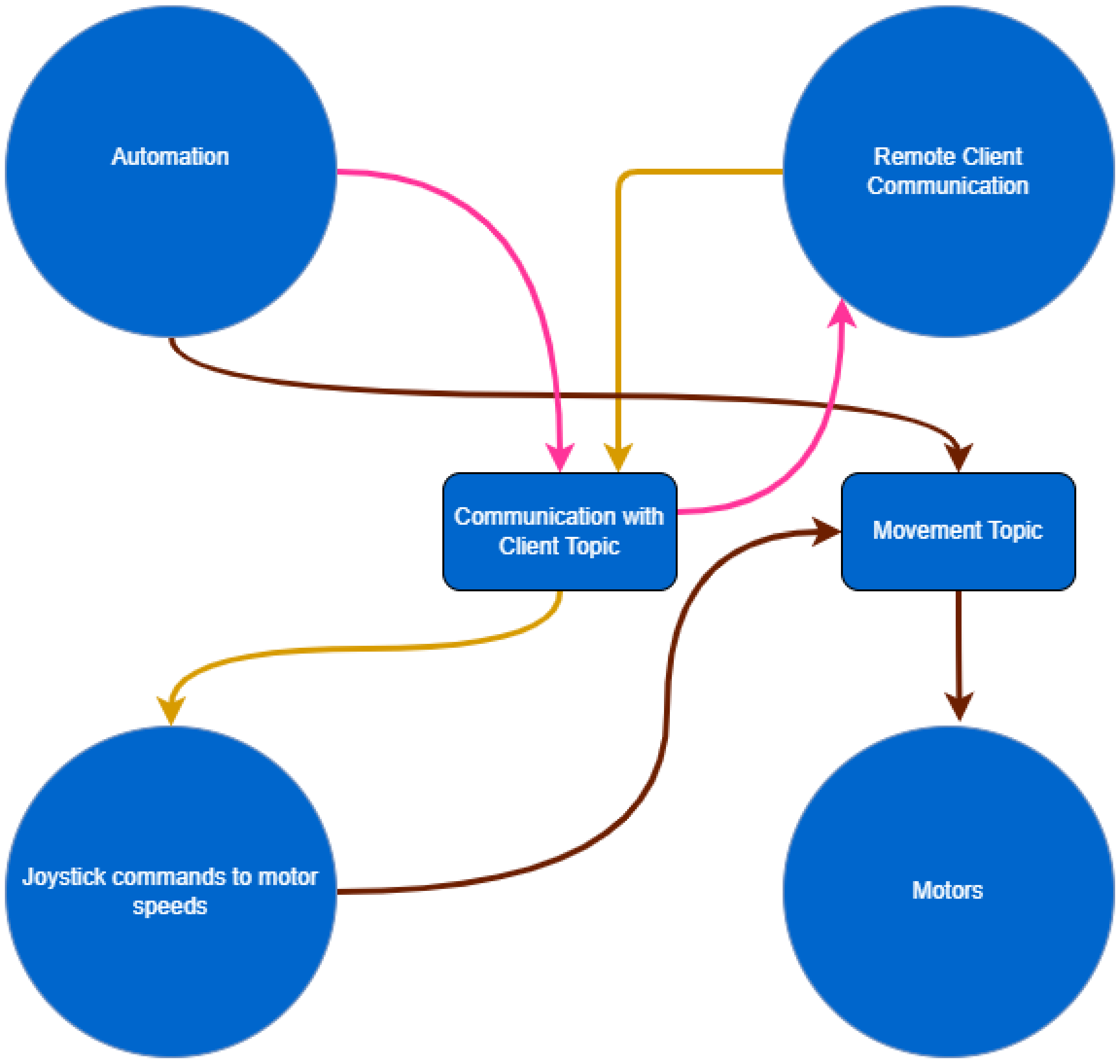
Automation
Remote Client Communication
Communication with Client Topic
Movement Topic
Joystick commands to motor speeds
Motors

Figure 4 - A high-level ROS2 graph of the robot's nodes

Firstly, we will look at the RCC node. The primary purpose of this abstracted node is to have the robot accept input from a joystick that is connected to the robot. Another purpose of the RCC node is to broadcast the video stream from the robot's ZED2i camera, whose data is in the automation node, onto a device with a GUI so that the remote client can see what the robot sees when the robot is operating. We will split these functionalities of the RCC node between two nodes, one which will accomplish the joystick communication task, and another node to broadcast the video (illustrated in Figure 5). This breakdown is to follow the modularization principles of ROS2 development.

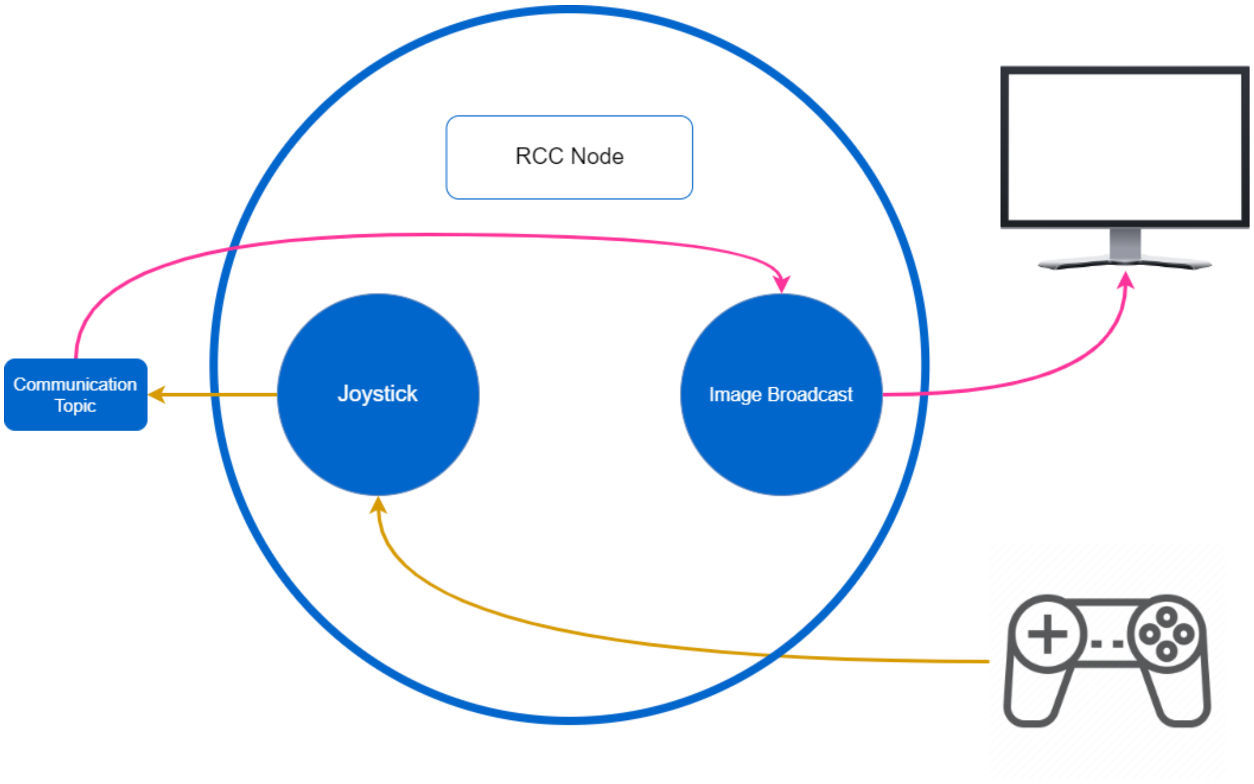

Figure 5 - RCC node breakdown

Secondly, we have the "Joystick Commands to Motor Speeds" node, which serves one purpose: to translate the joystick input signals into motor speed data, which is then passed along to the motor using the "Communication" topic.

Next, we will look at the "Motors" node, which is highly abstracted as the robot will have various kinds of motors, each with a different purpose. For example, some motors will power the wheels, whereas others will power the mechanism to mine and extract the regolith. Each of these motors will get a node, which will use the information from the "Movement" topic to do its respective task. These nodes are put together to form the overarching "Motors" node that is illustrated in Figure 4.

Finally, we have the automation node, which is also the most complex node to implement. It will consist of three nodes (illustrated in Figure 6) that work together to create the robot's autonomous functionality.

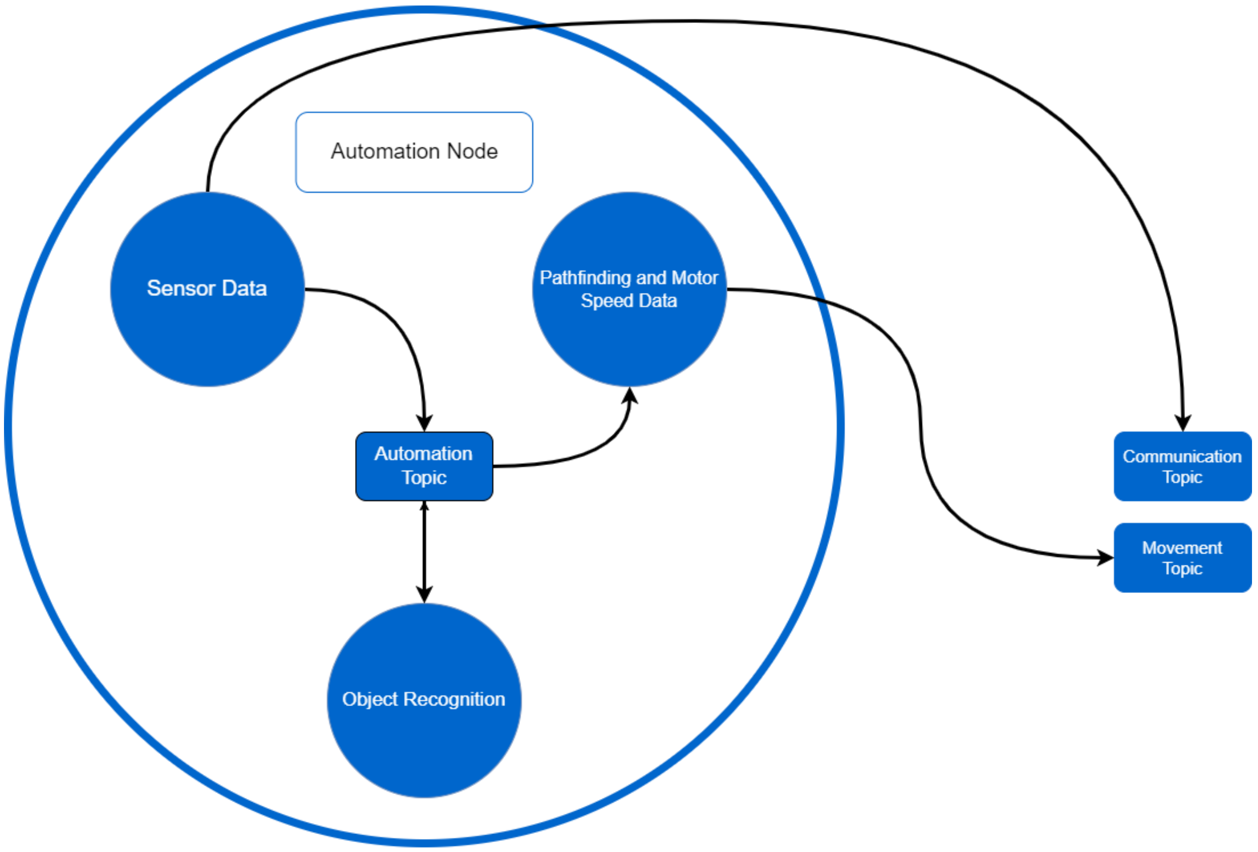

Figure 6 - Automation node breakdown

The "Sensor Data" node illustrated in Figure 6 will primarily deal with handling raw input data from the ZED2i camera and other sensors and publish it to the "Automation" topic. The node will also transmit the video frames as a message to the "Communication" topic so that the robot may broadcast the video to the device with a GUI. Concurrently, the "Object Recognition" node will implement the YOLOv3 object detection model that was discussed earlier to enable our robot to recognize obstacles and key regolith using the images and sensor data. The node will then publish the results from the model into the "Automation" topic. Subsequently, the "Pathfinding and Motor Speed Data" node will calculate the best action for the robot to take using information that it gathers from the "Automation" topic. The robot will have two main actions: move or dig. The dig action will only be called when the robot encounters the key regolith and if the regolith storage is not full. Otherwise, it determines the best way to move based on a combination of surrounding terrain of least resistance and the likelihood of regolith which is determined using the object recognition node data that is pushed into the "Automation" topic. Furthermore, this node will implement Dijkstra's pathfinding algorithm so that the robot can traverse the competition ground to drop off the regolith efficiently from any position on the competition ground.

As the breakdown above shows, we will have many nodes, each of which is critical to the functionality of the robot. To ensure that all the nodes function optimally, we will do thorough unit and integration testing on all nodes as we program them to minimize errors during the competition.

4.4 Risks

Risk	**Explanation & Risk Reduction**
Lack of Communication	Since this project has a very large separated team, any lack in communication can lead to slow progress. To increase our inter-team communication and avoid miscommunication, we started the project off with a Teams channel dedicated to open-communication and weekly check-ins. We also communicate task scheduling through the Razorbotz Trello board. The combination of the Teams channel, the Trello board, the Razorbotz discord server, and consistent emails/meetings with our supervisor (Andrew Burroughs), we can reduce communication risks.
Mechanical Failure	There are many points of possible mechanical failure related to the physical construction of the robot. Some major points of failure include the motors and motor controllers with possible short circuits. To reduce the chance of these risks, the MEEG team is using different motors & motor controllers which have a lower failure rate, and we will avoid any programmed angles/moves that might overload any components.
Differences in physical and simulation	In previous years, there have been issues with the coded tests not translating to the physical robot well. To reduce this risk, we are planning to start earlier and work in better conjunction with the MEEG team to understand their CAD designs as well as the physical robot. This allows us to notice any differences between CAD designs and the physical robot when making the autonomous program.
Library/Framework Version Mismatch	With the differences in member operating systems and updates in necessary libraries & frameworks, we may easily run into version control issues. To reduce the risk of this, we are comparing our ROS2 install procedures with previous years and getting all the installations done as early as possible. We also use the virtual machine VirtualBox when possible to unify our installation and coding processes.
Autonomous Failure	When we start programming for autonomous control, there are many risks that can lead to autonomous failure including failed object detection. We can reduce the effects of any autonomous failures

	through a “Kill Switch” button on the robot to shut off power and possible manual override to prevent a serious crash remotely.

4.5 Tasks

1.0 Researching competition

- 1.1 Understand objectives of the competition
- 1.2 Understand our responsibilities as the CSCE team
- 1.3 Meeting with supervisor to clarify questions and concerns regarding the CSCE team’s responsibilities

2.0 Learn ROS2

- 2.1 Install on computers
  - 2.1.1 Install VirtualBox (if not using Linux or M1 mac)
  - 2.1.2 Run Linux Distro
  - 2.1.3 Install ROS2 in the Linux OS
- 2.2 ROS2 Python & C++ tutorials

3.0 Review previous code

- 3.1 Understand what different components are doing
- 3.2 Understand when C++ vs Python is required

4.0 Streaming from the camera

- 4.1 Investigate potential solutions to enable camera streaming

5.0 General control

- 5.1 Implement nodes to control desired motor
  - 5.1.1 Will subscribe to manual/autonomous nodes that publish motor speeds

6.0 Manual control

- 6.1 Translating joystick commands into motor speeds
  - 6.1.1 Program motors to drive
  - 6.1.2 Program motors to begin mining process
  - 6.1.3 Program motors to dump mined materials
- 6.2 Communication with Client

6.2.1 Implement communication so controls can be sent to robot

6.3 Manual control nodes testing

6.3.1 Create unit tests for each node to ensure proper functionality

6.3.2 Create integration tests to ensure overall proper functionality

7.0 Autonomous control

7.1 Driving autonomy

7.1.1 Create object recognition nodes with ZED 2i stereo camera

7.1.1.1 Implement node for camera to publish images

7.1.1.1.1 Send robot's image stream to GUI

7.1.1.2 Implement node to receive images and handle object detection

7.1.2 Create nodes for the robot's path planning

7.1.2.1 Implement node to create path based on object detection

7.1.2.2 Implement node to translate path into motor speeds

7.1.3 Communicate motor speeds to the desired motors

7.2 Excavation Autonomy

7.2.1 Expected operations include:

7.2.1.1 Mine into the crater

7.2.1.2 Collect the resources mined

7.2.1.3 Store collected resources into temporary bucket

7.2.2 Implement algorithm that performs the expected operations for the excavation tool

7.2.2.1 Create node to control each motor based on the algorithm

7.3 Dumping Autonomy

7.3.1 Utilize object recognition and path planning nodes to detect and navigate to dump site

7.3.2 Implement node to control the position of the bucket for dumping once at dump site

7.4 Autonomous control nodes testing

7.4.1 Create unit tests for each node to ensure proper functionality

7.4.2 Create integration tests to ensure overall proper functionality

4.6 Schedule

Tasks	Assignee(s)	Dates
Researching competition	CSCE Team	10/1-11/27
Install ROS2 on personal machines	CSCE Team	10/10-10/17
Complete ROS2 Python tutorials	CSCE Team	10/17-10/24
Complete ROS2 C++ tutorials	CSCE Team	10/17-10/24
Investigate potential solutions for the video stream for the camera	CSCE Team	10/24-11/7
Review and understand code from previous years to ensure smooth transition into next semester	CSCE Team	11/1-11/15
Implement nodes to control desired motor	Jackson Newman Jackson Burger	1/17-1/23
Translating joystick commands into motor speeds	Justin Kilgo	1/17-1/23
Communication with client	Ahmed Moustafa Rohit Kala	1/17-1/23
Create unit and integration tests for each node for manual control	Justin Kilgo	1/23-1/30
Implement node for camera to publish images	Jackson Burger Ahmed Moustafa	1/23-1/30
Communicate motor speeds to desired motors	Jackson Burger Ahmed Moustafa	1/30-2/6
Implement node to receive images and handle object detection	Justin Kilgo Rohit Kala Jackson Newman	1/30-2/13
Create nodes for the robot’s path planning	Ahmed Moustafa Rohit Kala Justin Kilgo	2/6-2/20
Implement algorithm that performs expected operations for the excavation tool	Jackson Burger Jackson Newman	2/6-2/20

Tasks	Assignee(s)	Dates
Create node to control each motor based on algorithm	Ahmed Moustafa Justin Kilgo	2/20-2/27
Utilize object recognition and path planning nodes to navigate to the dump site	Justin Kilgo Rohit Kala	2/20-3/6
Implement node to control the position of the bucket for dumping once at dump site	Ahmed Moustafa Jackson Burger	2/27-3/13
Create unit and integration tests for each node of autonomous control	Ahmed Moustafa Rohit Kala Jackson Newman	3/13-4/3

4.7 Deliverables

- Project Website
  - We will create a page on the Capstone website for our project which will contain our project information, tasklist, and proposals.

- Project Proposal/Report
  - We will be creating a proposal for the original idea of how we are going to carry out the development of this project. Upon completion, we will create a report detailing our solution for the project. There will be clear indications of decisions that strayed away from our project proposal. We will also include the difficulties that we have faced and the process we took to overcome them.

- Autonomous Code
  - We will be programming the robot such that it is fully autonomous and will be able to accomplish the goals set by the NASA robot mining competition. To achieve this we will need to implement path planning, depth perception, and object recognition to detect obstacles. This deliverable will include the multiple ROS2 nodes for publishing images from the camera, taking those images and performing object detection, and planning the path to correctly publish the motor speed data.

- User Control Code
  - We will need to create a robot that can also be user controlled using a joystick. This will enable the robot to navigate the map and excavate material normally but will have a simulated five-second delay to resemble the long-distance communication from Earth to the Moon. We will need to program the client/server connection to allow us to send joystick inputs to the robot from a controller via our ROS2 code and have the robot respond accordingly.

- Robot Testing Data
  - The lunabot will make use of images to help it identify when a rock is in its path. It is important that this compiled dataset of images is varied enough to avoid overfitting for our object detection model.

5.0 Key Personnel

Ahmed Moustafa - Moustafa is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I, Programming Foundations II, Programming Paradigms, Software Engineering, Artificial Intelligence, and Algorithms. He is currently a Software Development Engineer Intern at SupplyPike working on full stack web development. Last summer, he attended the NACME-Google Applied Machine Learning Bootcamp working on a Reinforcement Learning model for a scale self-driving car. As a member of the CSCE portion of this project, he will be working on robot automation.

Jackson Burger - Burger is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Challenges I, Programming Challenges II, Programming Paradigms, Software Engineering, and Database Management Systems. He is currently an Annual Engineering & Technology Intern at J.B Hunt where he does full stack web development. As a member of the CSCE portion of this project, he will be working on robot automation.

Jackson Newman - Newman is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I, Programming Foundations II, Programming Paradigms, Database Management Systems, Software Engineering, and Algorithms. He is currently an annual Engineering and Technology intern for J.B. Hunt and works with the Continuous Integration and Continuous Development team. As a member of the CSCE portion of this project, he will be working on robot automation.

Justin Kilgo - Kilgo is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations II, Programming Paradigms, Software Engineering, and Algorithms. He has experience programming robots from his high school robotics team and was a Software Development Engineer Intern for Amazon over the summer working on Full Stack Web Development. As a member of the CSCE portion of this project, he will be working on robot automation.

Rohit Kala - Kala is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I, Programming Foundations II, Programming Paradigms, Software Engineering, Artificial Intelligence, and Algorithms. He is currently an undergraduate research assistant for the CVIU lab at UARK. As a member of the CSCE portion of this project, he will be working on robot automation.

Uche Wejinya - Dr. Wejinya is an associate professor in the Department of Mechanical Engineering at the University of Arkansas. He received his Ph.D. in Electrical Engineering from Michigan State University in August 2007. Dr. Wejinya's research relates to mechatronics with an emphasis on nanotechnology. His research delves into nanomaterials for nanosensors, nanoelectronics, and robotics which is relevant to this project.

6.0 Facilities and Equipment

- Facilities
    - Mechanical Engineering Robotics Laboratory - Lab in the Mechanical Engineering building where the robot is built and tested
    - Test Pit - Room in the Engineering Research Center used for larger tests that cannot be performed in the robotics lab.
- Equipment
    - ZED 2i stereo camera
    - SparkFun LIS2DH accelerometer
    - Jetson Nano processor
    - Pololu Glideforce GF23-120512-3-65 High-Speed LD Linear Actuator with Feedback: 12kgf, 12" Stroke (11.8" Usable), 3.3"/s, 12V
    - Falcon 500 Motors
    - USB to CAN converter

7.0 References

[1] Staff, The Robot Report. "Servi the Product of Bear Robotics and Softbank Robotics Partnership." The Robot Report, 29 Sept. 2020, https://www.therobotreport.com/servi-bussing-robot-product-bear-robotics-softbank-partnership/

[2] Spectrum, IEEE. "Roomba." *ROBOTS*, 18 May 2018, https://robots.ieee.org/robots/roomba/

[3] NASA, https://www.nasa.gov/sites/default/files/atoms/files/00_guidebook_2023_ver._0.3_tm.pdf.

[4] "Find Answers." *IRobot*, https://homesupport.irobot.com/s/article/19541.

[5] "Accelerometer Basics." *Accelerometer Basics - SparkFun Learn*, https://learn.sparkfun.com/tutorials/accelerometer-basics/all.

[6] "Triple Axis Accelerometer Breakout - lis2dh12 (Qwiic)." *SPX-15760 - SparkFun Electronics*, https://www.sparkfun.com/products/15760.

[7] *Figure 2: The Operation Principle of the Zed Camera.* https://www.researchgate.net/figure/The-operation-principle-of-the-ZED-camera_fig2_325854193.

[8] *How Does the Zed Work? – Help Center | Stereolabs*. https://support.stereolabs.com/hc/en-us/articles/206953039-How-does-the-ZED-work-.

[9] "Pytorch." *PyTorch*, https://pytorch.org/features/.

[10] *Survival Strategies for the Robot Rebellion*. https://pjreddie.com/media/files/papers/YOLOv3.pdf.

[11] Redmon, Joseph. *Yolo: Real-Time Object Detection*, https://pjreddie.com/darknet/yolo/.

[12] *CS106B: Programming Abstractions in C++*, https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1136/.

[13] "Ros 2 Documentation ." *ROS 2 Documentation - ROS 2 Documentation: Foxy Documentation*, https://docs.ros.org/en/foxy/index.html.

[14] "Object Detection." *Papers With Code*, https://paperswithcode.com/task/object-detection.

[15] Credit: Division of Rare and Manuscript Collections, et al. "Professor's Perceptron Paved the Way for AI – 60 Years Too Soon." *Cornell Chronicle*, 25 Sept. 2019, https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon.

[16] *MIT Deep Learning 6.S191*. http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf.

[17] "Training and Test Sets: Splitting Data | Machine Learning | Google Developers." *Google*, Google, https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[18] "What Is a Convolutional Neural Network?" *What Is a Convolutional Neural Network? - MATLAB & Simulink*, https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[19] "Common Objects in Context." *COCO*, https://cocodataset.org/.

[20] "Earth 'Will Expire by 2050'." *The Guardian*, Guardian News and Media, 7 July 2002, https://www.theguardian.com/uk/2002/jul/07/research.waste.

[21] Gilbert, Alex. "Mining in Space Is Coming." *Milken Institute Review*, https://www.milkenreview.org/articles/mining-in-space-is-coming.

[22] Kramer, Sarah. "Here's How Much Money It Actually Costs to Launch Stuff into Space." *Business Insider*, Business Insider, https://www.businessinsider.com/spacex-rocket-cargo-price-by-weight-2016-6.

[23] Magazine, Smithsonian. "The Robots That Paved the Way for Apollo." *Smithsonian.com*, Smithsonian Institution, 1 Feb. 2021, https://www.smithsonianmag.com/air-space-magazine/destination-moon-real-pioneers-180976726/.

[24] “Space Technologies at California.” *Home*, https://stac.berkeley.edu/project/rover#:~:text=An%20autonomous%20rover%20system%20deployed,these%20resources%20is%20incredibly%20difficult.

[25] Nayar, Jaya. “Not so ‘Green’ Technology: The Complicated Legacy of Rare Earth Mining.” *Harvard International Review*, Harvard International Review, 12 Aug. 2021, https://hir.harvard.edu/not-so-green-technology-the-complicated-legacy-of-rare-earth-mining/#:~:text=This%20stems%20from%20the%20fact,in%20especially%20detrimental%20health%20effects.

[26] GeeksforGeeks. “Client-Server Model.” *GeeksforGeeks*, 15 Nov. 2019, www.geeksforgeeks.org/client-server-model .

[27] “What Is TCP/IP? | Cloudflare.” Cloudflare, www.cloudflare.com/learning/ddos/glossary/tcp-ip/.