Autonomous Navigation of Mars Rover

Abhishek[1] Nikhil[2] Garima[3]

I. INTRODUCTION

The exploration of Mars has long been a significant goal for space agencies worldwide, driven by the planet's potential to reveal crucial insights about the history of our solar system and the possibility of past or present extraterrestrial life. Among the various missions, rovers have played a pivotal role due to their ability to traverse the Martian surface, conduct in-situ scientific experiments, and relay valuable data back to Earth. The advancement of autonomous navigation technology in these rovers is crucial to maximize their scientific output and operational efficiency. The Mars rover autonomous navigation project aims to enhance the rover's capability to navigate the complex and varied Martian terrain without human intervention. By developing sophisticated autonomous navigation systems, the rover can make real-time decisions, avoid hazards, and optimize its path to scientific targets, thereby increasing the overall success of the mission.

II. OBJECTIVE

The primary objective of the Mars Rover autonomous navigation project is to develop and implement advanced algorithms and systems that enable the rover to navigate the Martian surface autonomously. Specific goals include:

A. Obstacle Detection and Avoidance

Develop robust obstacle detection and avoidance mechanisms using yolo,to prevent the rover from getting stuck or damaged. Sensors like cameras, lidar, or sonar help rover perceive their surroundings and identify potential obstacles. Based on this information, the rover can make decisions to stop, change course, or maneuver around objects to safely reach its destination.

B. Terrain Analysis and Mapping

Develop systems to analyse and map the Martian terrain accurately. This includes identifying and categorising various surface features like rocks and craters. This information is used to create maps that help rovers navigate safely, avoid obstacles, and plan efficient routes to reach interesting scientific locations.

By achieving these objectives, the project aims to significantly enhance the Mars rover's operational capabilities, allowing for more extensive and productive exploration of the Martian surface, ultimately contributing to our understanding of Mars and the broader objectives of planetary science.

III. OBJECT DETECTION AND SIZE ESTIMATION

Computer vision means extracting information from images, text, videos. OpenCV, a cross-platform and free-to-use library of functions, is based on real-time Computer Vision, which supports Deep Learning frameworks that aid in image and video processing.

- Object Detection:The location of the object.
- Object Recognition:The objects in the image and their position.
- Object Classification:The broad category that the object lies in.
- Object Segmentation:The pixels belonging to that object.

A. Edge Detection

1) Theory: Edge detection is a traditional technique for finding the boundaries of objects in an image. It works by detecting sharp discontinuities in brightness.

- Image Smoothening
- Sobel filter applied to detect edges
- Non-Max suppression
- Thresholding
- Hysteresis

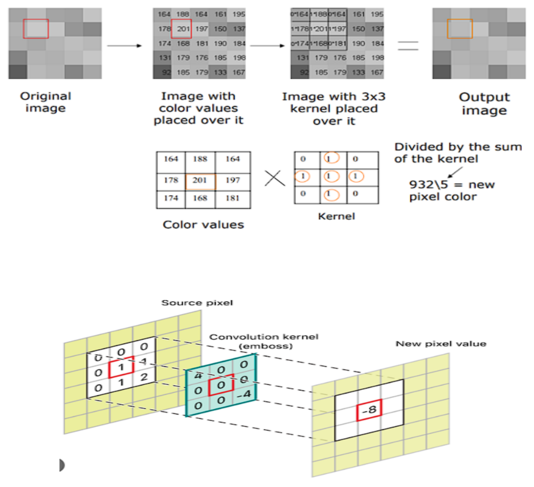

Original image
Image with color values placed over it
Image with 3x3 kernel placed over it
Output image
Color values
Kernel
Divided by the sum of the kernel
932\5 = new pixel color
Source pixel
Convolution kernel (emboss)
New pixel value
-8

Fig. 1. **Convolution of kernel over an image**

A convolution is done by multiplying a pixel's and its neighboring pixels color value by a matrix. Kernel is a matrix whose number are like weights multiplied to the image matrix such that value of center pixel can be replaced. We first smooth the input image by convolving it with a Gaussian kernel. Gaussian Kernel is usually used to blur an image.

The equation for a Gaussian kernel of size (2k+1)×(2k+1) is given by:

$$h_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-2(i-k)^2 + (j-l)^2\right), 0 \leq i, j < 2l+1 \tag{1}$$

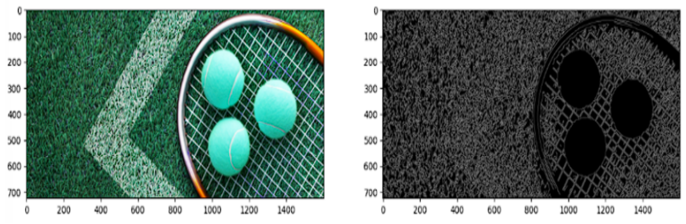


Fig. 2. **Output of Canny Image**

Now, we want to get some features, such as circles, lines, etc., from the Edge Image we have received after Edge Detection. This is where Hough transform comes in. The Hough transform is a **feature extraction technique** used in image analysis, computer vision, and digital image processing. The technique aims to find imperfect instances of objects within a certain class of shapes by a **voting procedure**.

- Determine the range of and . Typically, the range of is [0, 180] degrees, and is [-d, d], where d is the diagonal length of the edge. Therefore, it's crucial to quantify the range of and , which means there should only be a finite number of potential values.
- Create a 2D array called the accumulator with the dimensions (num rhos, num thetas) to represent the Hough Space and set all its values to zero.
- Use the original image for edge detection (ED).
- Check each pixel on the edge picture to see if it is an edge pixel. If the pixel is on edge, loop over all possible values of , compute the corresponding , locate the and index in the accumulator, and then increase the accumulator base on those index pairs.

Iterate over the accumulator's values. Retrieve the and index and get the value of and from the index pair. If the value exceeds a specified threshold, we can then transform the index pair back to the form of y = ax + b..

2) Implementation: The mentees were assigned their first task of the project which dealt with the detection of tennis ball in any random image. We applied the Canny edge

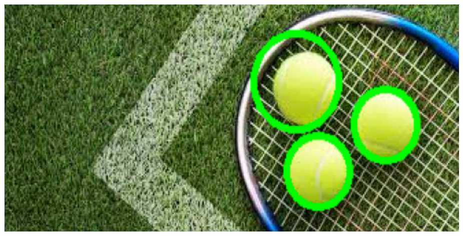


Fig. 3. **Output Image**

detection algorithm to identify strong edges in the image. This reduced the image data to focus on areas with significant intensity changes, which are likely to be object boundaries. Using the Hough transform we were able to accumulate the curves in a parameter space, which allowed us to identify the most likely circles based on their prevalence. Finally we had identified the potential locations of tennis balls in the image based on the circles found in the Hough transform.

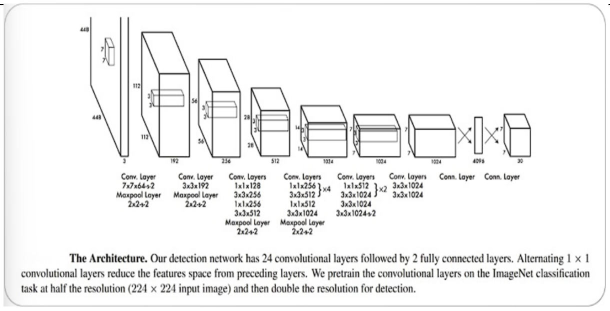
3) Conclusion: We covered the fundamental aspects of computer vision, emphasizing its role in extracting information from images, text, and videos through the use of OpenCV, a versatile library for real-time computer vision tasks. We delved into object detection, recognition, and classification, highlighting traditional techniques such as edge detection and image smoothing with Gaussian kernels. Additionally, we discussed the Sobel filter for edge detection, non-max suppression, thresholding, hysteresis, and feature extraction techniques like the Hough transform. These combined methods create a comprehensive approach to improving the rover's ability to navigate and explore challenging terrains on Mars.

B. Object Detection using YOLO v7 on a Custom Dataset

1) Theory: **Object detection** is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. During the project, mentees were briefed about object detection using the YOLO v7 algorithm and assigned a task to implement the algorithm to detect objects. **You Only Look Once (YOLO)** algorithm proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities simultaneously. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin. YOLO v7 is one of its most popular versions. The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image.The architecture of the CNN model that forms the backbone of YOLO is shown below. One key technique in the YOLO models is **on-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

2) Implementation: The mentees were assigned with this second task to train a YOLO v7 model using a custom dataset. The task involved downloading a custom dataset of Rock-Paper-Scissors from Roboflow and using that dataset to train our YOLO model. To train our detector, we took the following steps:

- Install YOLOv7 dependencies


448
448
112
112
56
56
28
28
14
14
7
7
3
192
256
512
1024
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

Fig. 4. **Architecture of YOLO**

- Load custom dataset from Roboflow in YOLOv7 format
- Run YOLOv7 training
- Evaluate YOLOv7 performance
- Run YOLOv7 inference on test images

First, we downloaded the YOLO repository and installed the requirements. Next, we downloaded our dataset in the right format. We used the YOLOv7 PyTorch export. Note that this model required YOLO TXT annotations, a custom YAML file, and organized directories. The Roboflow export wrote this for us and saved it in the correct spot. After starting the custom training, we modified the epochs to a hundred. YOLO divides an input image into an S × S grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is. YOLO gives us the centroid of objects by localizing them in a contour or frame. Using the centroid, we can extract the depth of that point using the point cloud.

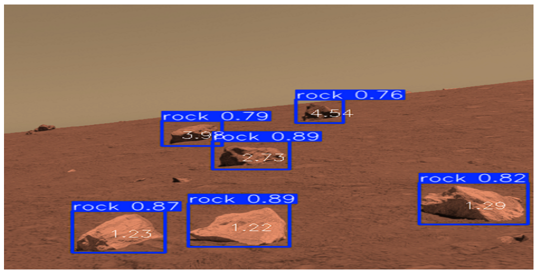


Fig. 5. **Detection of Rocks using YOLO**

3) Conclusion: In conclusion, we have explored the implementation and training of the YOLO v7 algorithm for object detection, focusing on its application to detect objects in images using a deep convolutional neural network. We learned to train a YOLO v7 model using a custom dataset and perform essential steps such as installing dependencies, loading datasets, running training, evaluating performance, and making inferences. To summarize, we tried improving obstacle detection and avoidance for the Mars rover by using YOLO and sensors such as cameras. By incorporating these technologies, the rover can effectively identify and navigate around obstacles, ultimately enhancing its operational capabilities for more successful exploration.

IV. Depth Estimation and SLAM

A. Extraction of depth maps

1) Theory: ZED camera is a type of stereo camera designed for capturing depth information. Unlike traditional cameras that capture a flat image, the ZED camera uses two lenses to mimic human vision and perceive depth. This allows it to create 3D models of its surroundings. ZED cameras are used in various applications that require depth perception, such as robotics, virtual reality, and object recognition. Point cloud is a collection of data points in 3D space. Imagine taking a bunch of tiny dots and sprinkling them around a scene to represent its shape and features. Each point has information about its location, and some may also include color or intensity data. Point clouds are like a raw sketch of a 3D scene captured by sensors like LiDAR scanners. Depth maps are like grayscale images that reveal how far away objects are from a camera. Imagine a regular picture, but instead of showing colors, different shades of gray represent distance. Lighter areas correspond to closer objects, while darker areas depict objects farther away. Depth maps are often created by special cameras or sensors that can capture distance information along with the visual scene.

2) Implementation: The third task was based on extraction of depth of martian terrain and . This was initiated with training the YOLO model to detect rocks in Mars images. Once an image is processed, the model outputs bounding boxes, class labels, and confidence scores for the detected objects. Initially, the bounding box coordinates are extracted to find the centroid of the rock by averaging the x and y coordinates. Then, using the provided depth image, the depth values within the bounding boxes are extracted and their average is normalized between 0 and 1. Based on the normalized depth value, the relative depth is classified into predefined categories such as very close or far away. The image below shows the extraction of point cloud and depth maps using the ZED camera.

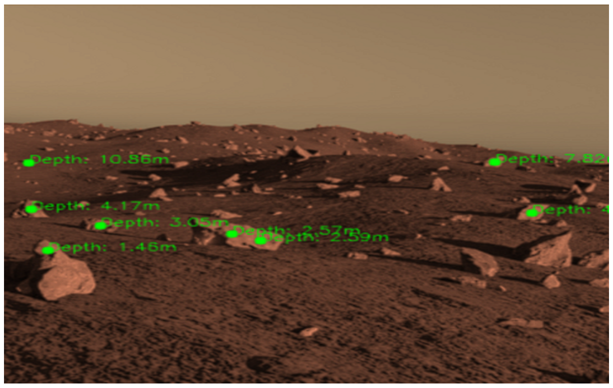


Fig. 6. **Depth map of rocks**

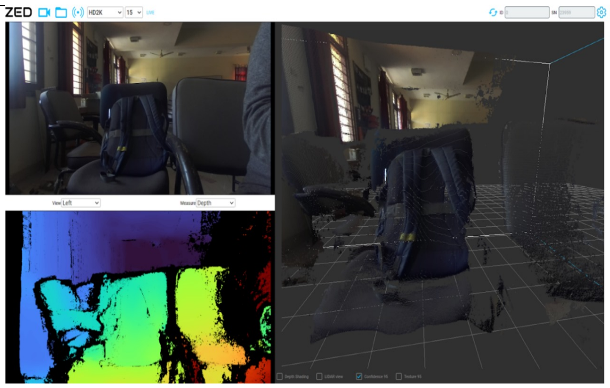


Fig. 7. **Simulation in ZED Camera**

3) Conclusion: We examined the use of depth estimation techniques, particularly with the ZED camera, to capture and utilize depth information in 3D space. By integrating point clouds and depth maps, we enhanced object detection capabilities, enabling the accurate classification of objects based on their relative depth.

B. Simulation and Mapping with ROS, Gazebo and SLAM

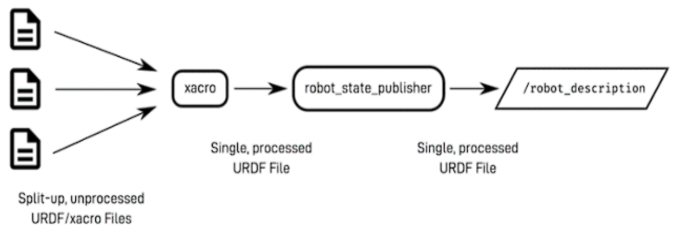
1) Theory: ROS 2 is an open-source framework designed to facilitate the development of robot software. It has the advantages of:

- Real-time capabilities
- Multi-robot systems
- Security
- Cross-platform Support
- DDS(Data Distribution Service)

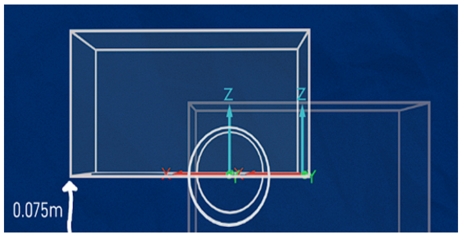
Eventually we learnt the basics of ROS through turtlesim.Gazebo is a powerful, open-source robot simulation tool that integrates seamlessly with ROS. It allows developers to simulate robots in complex indoor and outdoor environments.Gazebo offers:

- 3D simulation
- Physics engine
- Sensor Simulation
- Extensibility
- Integration with ROS

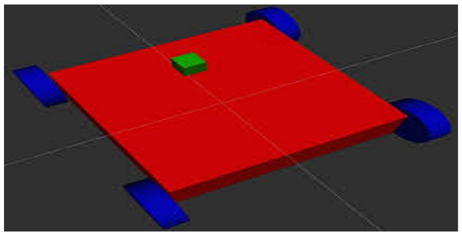
URDF(Unified Robot Description Format):An XML format used to describe the physical configuration of a robot,including its links, joints, and sensors. Xacro lets us manipulate the URDF files in various ways, but the two main ones are splitting up code into multiple files and avoiding duplicate code.


xacro
robot_state_publisher
/robot_description
Single, processed URDF File
Single, processed URDF File
Split-up, unprocessed URDF/xacro Files

- Base link- It is the main coordinate frame for the robot. Other links will be connected to the base link through different types of joints
- Chassis- a simple box of dimension 0.3 × 0.3 × 0.15m. The reference point of the chassis will be at the rearview centre; hence, the chassis link will be connected to our base link through the fixed joint, set back a little from the centre. By default, the box geometry will be centred around the link origin. We want the link origin at the rear bottom of the box, so to achieve that, we shift the box forward (in X) by half its length (0.15m) and up (in Z) by half its height (0.075m).



- Drive wheels- The drive wheels will move continuously; hence, it is connected to the base link via a continuous joint
- Caster wheels- We've added a frictionless sphere connected to our chassis with its lowest point aligning with the base of the wheels.



Use RViz (ROS visualization tool) to visualize sensor data and robot state in real time. Adding collision and inertia-

- Collision- This is used for physics collision calculations. We can set the geometry and origin.
- Inertia- This is also used for physics calculations but determines how the link responds to forces. The inertial properties are:
  - Mass- mass of the link
  - Origin- the centre of mass
  - Rotational inertia matrix

Simultaneous Localization and Mapping (SLAM) is a technique used for mobile robot to build and generte a map from the environment it explores (mapping). The generated map then is used to determine the robot and surrounding landmark location and make an appropriate path planning for the robot (localization). The process of mapping and localizing in SLAM is done concurrently where the mobile robot relatively creates the map. The created map is used to calculate and estimate landmark position and mobile robot trajectory. SLAM advantage is that it is able to generate geometrically consistent environment map and localized robot position and landmark concurrently. These became the major factors which makes SLAM the technique in autonomous mobile robot field. Some of the key aspects of SLAM are listed below.

- Localization: Determining the robot's position within the environment.
- Mapping: Building a map of the environment based on sensor data.
- Sensor Fusion: Combining data from multiple sensors to improve accuracy.

Nav2 (Navigation 2) is a ROS 2 package providing tools and algorithms for autonomous navigation.

- Setup: Launching with robot model SLAM and Nav2 nodes to start Gazebo stimulation and configuring the parameters to match our robot and environment, such as scan topics, map resolution, localization accuracy, and navigation goals.
- Visualize the Map and Navigation: Use RViz to visualize the real-time map being built by the SLAM algorithm and monitor the robot's navigation path.

2) Implementation: In this project we are going to use ROS as it helps robots navigate their environments and provide the structure and tools for different SLAM algorithms to communicate and share data, allowing the robot to build a map while keeping track of its location within that map. The next task for the mentees was to understand the ROS applications and get involved in making a simple subscriber-publisher node and try out simulating the URDF file through Gazebo. Now, let us have a look at the fundamental imple-

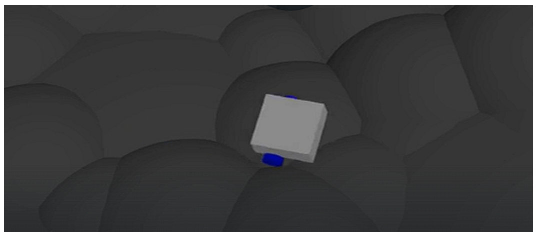


Fig. 8. **Output Image of Simulation in Gazebo**

mentation of the SLAM. Suppose a robot exploring through, using a sensor device to observe and estimate the landmark location in the environment.The figure **(Fig.10)** below shows the SLAM state of art. The robot uses sensors like cameras, LiDAR (Light Detection and Ranging), or ultrasonic sensors to gather data about its environment. This data includes things like distances to obstacles, landmarks, and other features. Based on the sensor data and its estimated location, the robot updates its internal map of the environment. This map can be a simple representation of walls and obstacles, or a more complex one that includes details like furniture or specific landmarks. Using the updated map and new sensor data, the robot refines its understanding of its current location within the environment.This allows it to navigate safely and efficiently. SLAM relies on several mathematical concepts to function:

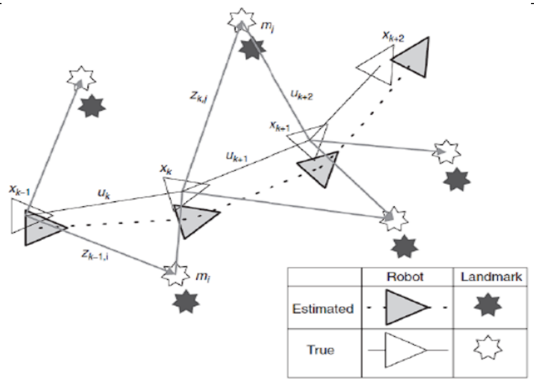


Fig. 9. **SLAM state of art**

- Probability and Statistics: Since sensor measurements can be noisy or imprecise, SLAM algorithms often use probabilistic techniques to represent the robot's location and the map with varying degrees of certainty.
- Linear Algebra: Matrices and vectors are used to represent the robot's pose (position and orientation), sensor measurements, and the map itself. Mathematical operations on these elements allow the robot to update its map and location based on new information.
- Kalman Filters: A popular technique for handling noisy sensor data and estimating the robot's state (location and pose) is the Kalman filter. This filter helps the robot weigh new sensor measurements against its existing map to provide the most likely estimate of its location.

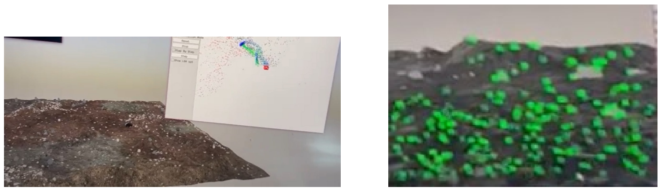


Fig. 10. **Output images from SLAM**

3) Conclusion: In summary, we have learned the basics of ROS 2, focusing on its real-time capabilities, multi-robot systems, security, and cross-platform support. We have practiced using turtlesim and explored Gazebo for 3D robot simulation, which seamlessly integrates with ROS. We have covered the fundamentals of URDF for robot description and the use of Xacro for efficient file management. We discussed key robot components such as base links, chassis, drive wheels, and caster wheels. Using RViz, we visualized sensor data and robot states in real-time and understood the importance of collision and inertia properties in simulations.

We explored the basics of SLAM (Simultaneous Localization and Mapping), a crucial process for creating maps of unknown environments while tracking a robot's location. Key components of SLAM include localization, mapping, and sensor fusion to enhance accuracy. We installed and used SLAM-related packages such as slam toolbox, cartographer, and gmapping. Using Nav2 for autonomous navigation,we configured Gazebo simulations and adjusted parameters to match our robot and environment. Finally, we utilized RViz to visualize the real-time map and track the robot's navigation path, deepening our grasp of SLAM and autonomous navigation in robotics.

V. Conclusion

In summary, we enhanced the Mars rover's ability to detect and avoid obstacles by integrating YOLO and cameras, improving its navigation and exploration capabilities. We also delved into computer vision basics using OpenCV, covering object detection and feature extraction methods. Furthermore, we trained a YOLO v8 model with a customized dataset and explored depth estimation using the ZED camera. We also learned the fundamentals of ROS 2, focusing on real-time capabilities, multi-robot systems, and cross-platform support. Additionally, we utilized Gazebo for 3D simulations and RViz for real-time visualization to understand key robot components and the significance of collision and inertia properties. Furthermore, we explored SLAM (Simultaneous Localization and Mapping) to create maps and track the rover's location, using packages such as slam toolbox and cartographer. By configuring Gazebo simulations and using Nav2 for navigation, we gained insights into autonomous navigation and SLAM in robotics. These methods collectively enhance the rover's ability to navigate and explore challenging terrains.