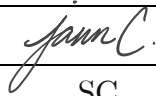




**Department Aerospace Engineering**  
**Faculty of Engineering & Architectural Science**

Course Code	AER 627
Section	02
Course Title	Introduction to Space Robotics
Semester/Year	Winter 2021
Instructor	John Enright
TA	Joel Moore
Project 4	Rover
Submission Date	April 16 <sup>th</sup> 2021

Name	Student ID	Signature*
Jann Cristobal	500815181	
Stephanie Chan	500819304	SC

*\*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:*  
<http://www.ryerson.ca/senate/policies/pol60.pdf>

## **Abstract**

In this project, the rover is able to go from the start position to the goal position. The large physical size of the rover contributes to its inability to completely go around obstacles. The path planning algorithm also require improvement as it tend to get close to the obstacles. The system prove to be robust under scan failures by using older data to try and find the goal.

## Table of Contents

Table of Contents .....	2
1 Introduction.....	3
2 Theory.....	3
2.1 Package A: Rover Controller (Jann Cristobal) .....	3
2.2 Package B: Path Planning (Stephanie Chan) .....	7
3 Design Documentation.....	8
3.1 Package A: Rover Controller (Jann Cristobal) .....	8
3.1.1 Physical Design.....	8
3.1.2 Control Theory .....	8
3.1.3 Implementation.....	10
3.2 Package B: Path Planning (Stephanie Chan) .....	12
4 Project Operations .....	13
5 Conclusion .....	16
6 References .....	17
7 Appendix .....	18
7.1 Engineering Drawing.....	18

## List of Figures

Figure 2.1: Differential Drive Rover Schematic .....	3
Figure 2.2: Rover pose at two different state.....	5
Figure 2.3: Basic Rover Control Schematic .....	6
Figure 3.1: Isometric View of the Rover .....	8
Figure 3.2: Closed-loop block diagram.....	9
Figure 3.3: Rover Control Flowchart.....	10
Figure 4.1: Actual view of the scene.....	14
Figure 4.2:Occupancy Map of the scene .....	14
Figure 4.3: Trajectory Generated from start to goal.....	15
Figure 7.1: Front view of the rover .....	18
Figure 7.2: Top view of the rover .....	18
Figure 7.3: Right view of the Rover .....	19

# 1 Introduction

In this project, the goal is to develop an effective controller for a differential-drive rover and a path-finding algorithm that is able to navigate through an environment filled with various obstacles.

## 2 Theory

This section outlines the mathematical theories utilized in designing and programming the robot.

### 2.1 Package A: Rover Controller (Jann Cristobal)

The rover that will be utilized in this project is a differential drive with a spherical wheel to keep the structure stable. The first step to designing a controller is to understand the various components and parameters that make up the rover.

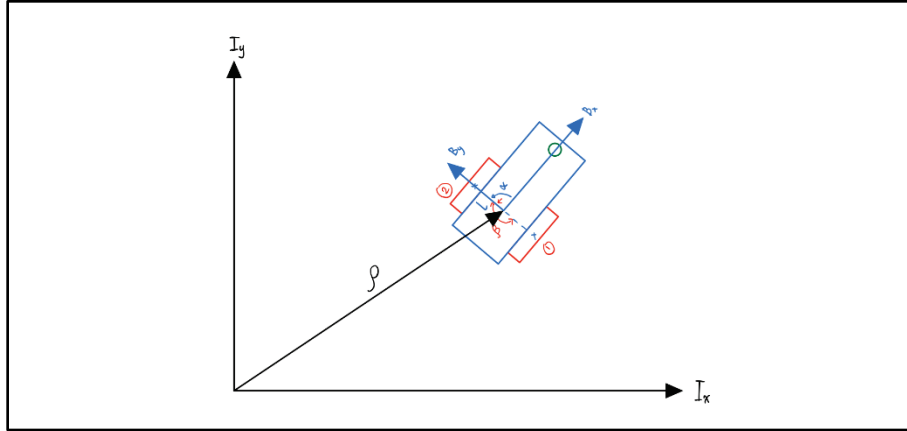


Figure 2.1: Differential Drive Rover Schematic

Figure 2.1 shows a top-down view of what the rover would look like with respect to the inertial frame of reference,  $I$ . In the body frame of the rover,  $B_x$  is the axis that points in the forward direction of motion, and  $B_y$  is the axis that points to the left direction, and  $B_z$  is the axis that points upward.  $\theta$  is called the heading, which is the angle between  $B_x$  and  $I_x$ .  $\vec{\rho}$  is the position vector of the rover with respect to the inertial frame. The goal is to determine  $\vec{v}_B^{B/I}$  and  $\vec{\omega}_B^{B/I}$ , then using the heading calculate the  $\vec{v}_I^{B/I}$  and  $\vec{\omega}_I^{B/I}$ . Knowing that each wheel does half of the work, and that we are primarily dealing with planar motion, the resulting forward kinematics is as follows.

$$\vec{v}_I^{B/I} = \vec{C}_{IB} \times \vec{v}_B^{B/I} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{2}R(\dot{\phi}_1 + \dot{\phi}_2) \\ 0 \\ 0 \end{bmatrix} \quad (2.1.1)$$

$$\vec{\omega}_I^{B/I} = \vec{C}_{IB} \times \omega_B^{B/I} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ \frac{R}{2L}(\dot{\phi}_1 + \dot{\phi}_2) \end{bmatrix} \quad (2.1.2)$$

Where  $L$  is half of the wheelbase which measures the distance from the origin of the vehicle frame and the wheels.  $R$  is the radius of the wheels.  $\dot{\phi}_1$  and  $\dot{\phi}_2$  are the wheel speeds, respectively.

For a differential drive, there are two standard wheels labeled 1 and 2 drawn in red, and a spherical wheel colored green in Figure 2.1.  $\alpha$  is the angle between the  $B_x$  axis and  $L$  vector.  $\beta$  is the angle between the  $B_y$  and the  $L$  vector.  $R$  is the wheel radius.  $\rho$  is the position vector relating the vehicle frame to the inertial frame. There are two main constraints associated with this type of drive system. Rolling constraints shown in equation 2.1.3 and lateral slip constraints shown in equation 2.1.4.

$$\vec{w}^T \vec{v} - R\dot{\phi} = 0 \quad (2.1.3)$$

$$\vec{u}^T \vec{v} = 0 \quad (2.1.4)$$

Where  $\vec{w}^T$  is shown in equation 2.1.5,  $\vec{v}$  is the vehicle velocity,  $R$  is the radius,  $\dot{\phi}$  is the wheel speed, and  $\vec{u}^T$  is shown in equation 2.1.7

$$\vec{w}^T = [\sin(\alpha + \beta), -\cos(\alpha + \beta), -L \cos(\beta)] \quad (2.1.5)$$

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \quad (2.1.6)$$

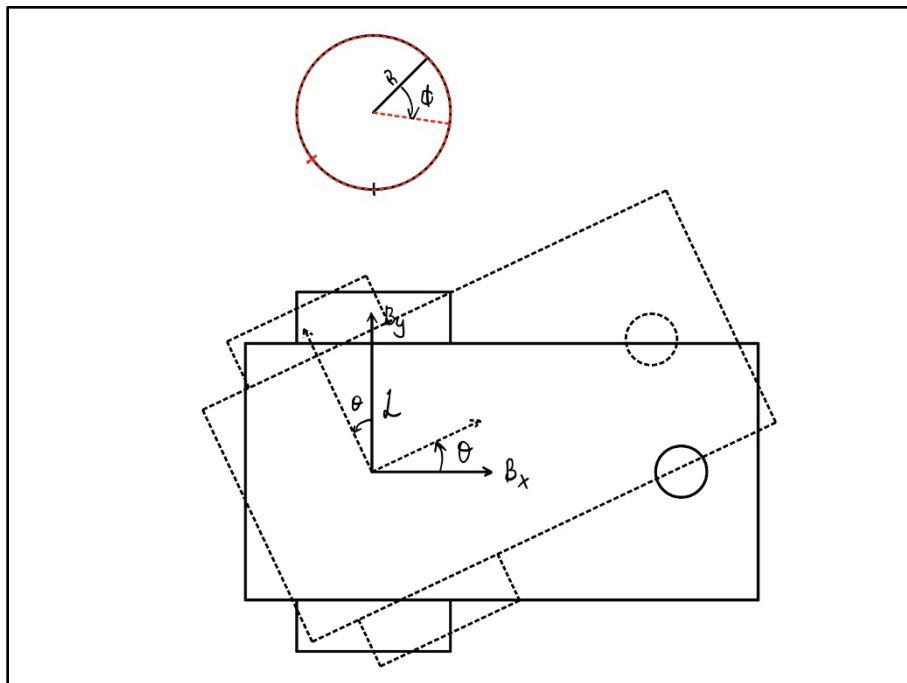
$$\vec{u}^T = [\cos(\alpha + \beta), \sin(\alpha + \beta), L \sin(\beta)] \quad (2.1.7)$$

The goal is to calculate the required wheel speed given the rover's speed and direction. After combining equations 2.1.5 to 2.1.7 the resulting inverse kinematics is

$$\begin{bmatrix} \vec{J}_1 \\ \vec{C}_1 \end{bmatrix} \vec{v} = \begin{bmatrix} \vec{J}_2 \dot{\phi} \\ \vec{\phi} \end{bmatrix} \quad (2.1.8)$$

$$\begin{bmatrix} \sin(\alpha_1 + \beta_1) & -\cos(\alpha_1 + \beta_1) & -L_1 \cos(\beta_1) \\ \sin(\alpha_2 + \beta_2) & -\cos(\alpha_2 + \beta_2) & -L_2 \cos(\beta_2) \\ \cos(\alpha_1 + \beta_1) & \sin(\alpha_1 + \beta_1) & L \sin(\beta_2) \\ \cos(\alpha_2 + \beta_2) & \sin(\alpha_2 + \beta_2) & L \sin(\beta_2) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \\ 1 \end{bmatrix} = \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ 0 \\ 0 \end{bmatrix} \quad (2.1.9)$$

For implementation, the controlling the motor's velocity is not as straight forward as controlling the vehicle's pose.

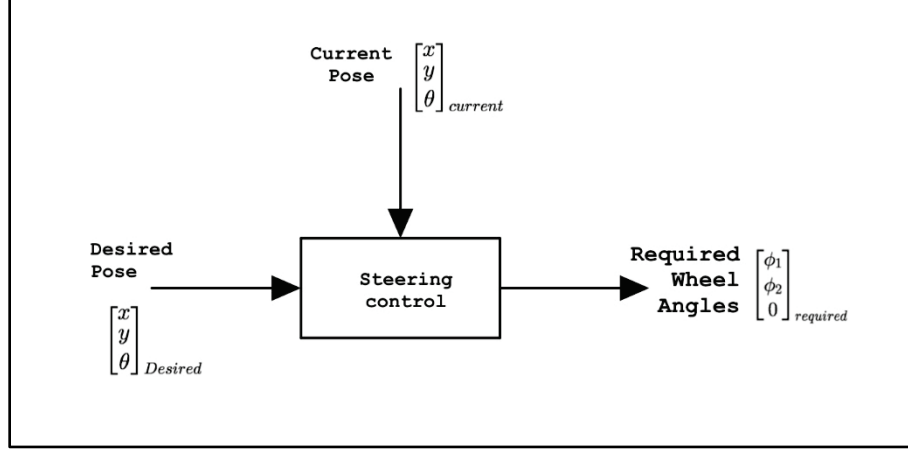


*Figure 2.2: Rover pose at two different state.*

Figure 2.2 shows the pose of the rover from state to another. And based on the assumption that each wheel does half of the work, Equation 2.1.10 shows how change in heading,  $\theta$ , relates to the wheel rotation,  $\phi$ .

$$\frac{R}{\phi} = \frac{L}{2\theta} \quad (2.1.10)$$

External vision sensor will be used to analyze the scene for both path planning and to determine the rover's current pose. The basic concept for how the rover will be controlled is shown in Figure 2.3



*Figure 2.3: Basic Rover Control Schematic*

The basic control schematic seen in Figure 2.3 is based on figure 4 shown in the project instructions [1]. The major difference is that instead of having linear and angular velocities as inputs, desired pose and current pose over a discretized time are used. Both methods work similarly, the reason for using location and heading as opposed to rover velocity is that it is easier to implement it with the Lego EV3. A detailed block diagram and pseudo code will be available in the design documentation section.

## 2.2 Package B: Path Planning (Stephanie Chan)



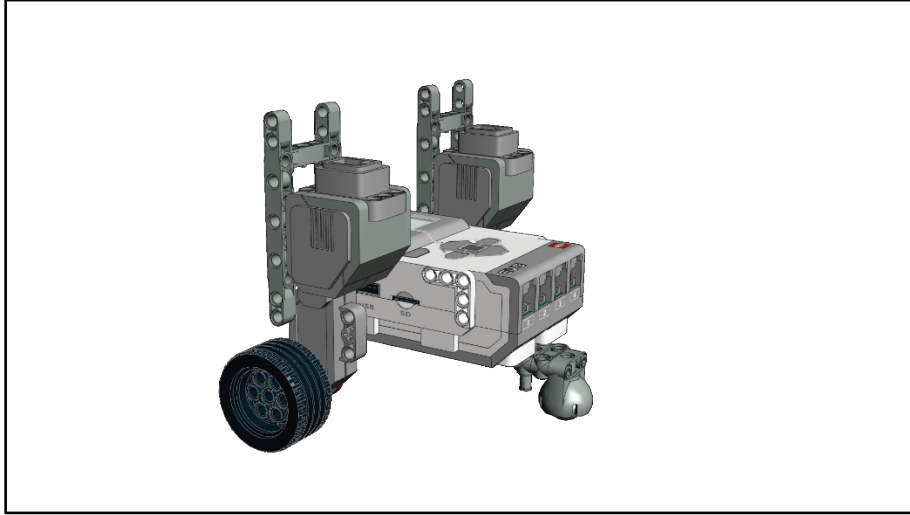
### 3 Design Documentation

This section will show the design process involve of building and programming of the rover.

#### 3.1 Package A: Rover Controller (Jann Cristobal)

##### 3.1.1 Physical Design

When it comes to the physical construction of the rover, there are three main concerns that must be considered. The height of the rover can affect the performance of the scans and must be kept about the same height as the rest of the obstacles. The visibility of the April tag for the vehicle will affect trajectory generation and must ensure that there is minimal obstructions. Lastly, the overall size of the rover must be kept as small as possible to represent the vehicle model more accurately in the simulation. The final design can be seen in Figure 3.1.



*Figure 3.1: Isometric View of the Rover*

The three view drawings can be referred to in 7.1 Engineering Drawing.

##### 3.1.2 Control Theory

To get to the desired location, there are two main aspect of the rover that must be controlled. The first one is the steering, and the second one is the forward motion. To control steering, the required heading must be calculated and related to required wheel rotation according to equation 3.1.1. This is proportional part of the controller. The required heading can be calculated by simply taking the difference between the current and desired heading.

$$\theta_{required} = \theta_{current} - \theta_{desired} \quad (3.1.1)$$

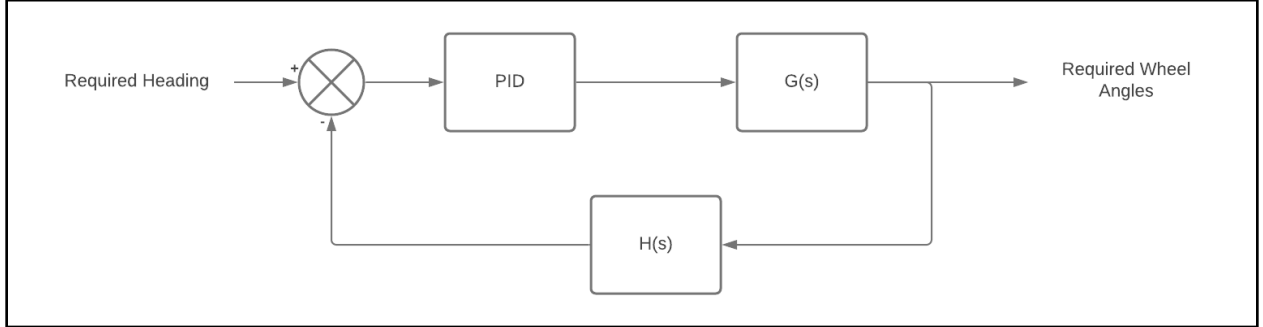
The required wheel angles can then be calculated as

$$\phi_{required} = \frac{2R\theta_{required}}{L} \quad (3.1.2)$$

Where  $R$  is the wheel radius,  $L$  is half the wheelbase. Note that the required wheel angles for both left and right wheel will be equal since the work is evenly distributed. To turn right, left wheel must rotate such that the rover moves forward, and the right wheel must rotate such that the rover moves backward. The opposite is true in order to turn left.

Once the rover heading accurately pointed, the rover must now move forward. To do this, the wheels must simply rotate so that the whole rover is translating in the  $B_x$  direction. From Figure 2.1, the wheel must rotate counterclockwise about the  $B_y$  axis. So far, our input is the required heading and output is the required wheel angles.

The open loop control is effective for ideal condition where the dynamic factors are not considered. For the actual system, a closed loop control is necessary to compensate for the errors that arise from external factors such as friction, bending, uneven surface, distortion, and overshoot. Figure 3.2 shows the closed loop block diagram for the system.



*Figure 3.2: Closed-loop block diagram.*

The  $G(s)$  is essential the transfer function relating the initial  $\phi_{required}$  and  $\theta_{required}$  from the open loop control. The  $H(s)$  is the resulting heading to the actual heading using the vision sensors. It is a negative feedback which allows for the errors to be correct by a PID controller. PID stands for proportional-plus-integral-plus-derivative controller [2]. In the s-domain, it is expressed as

$$G_c(s) = k_p + \frac{k_i}{s} + k_d s \quad (3.1.3)$$

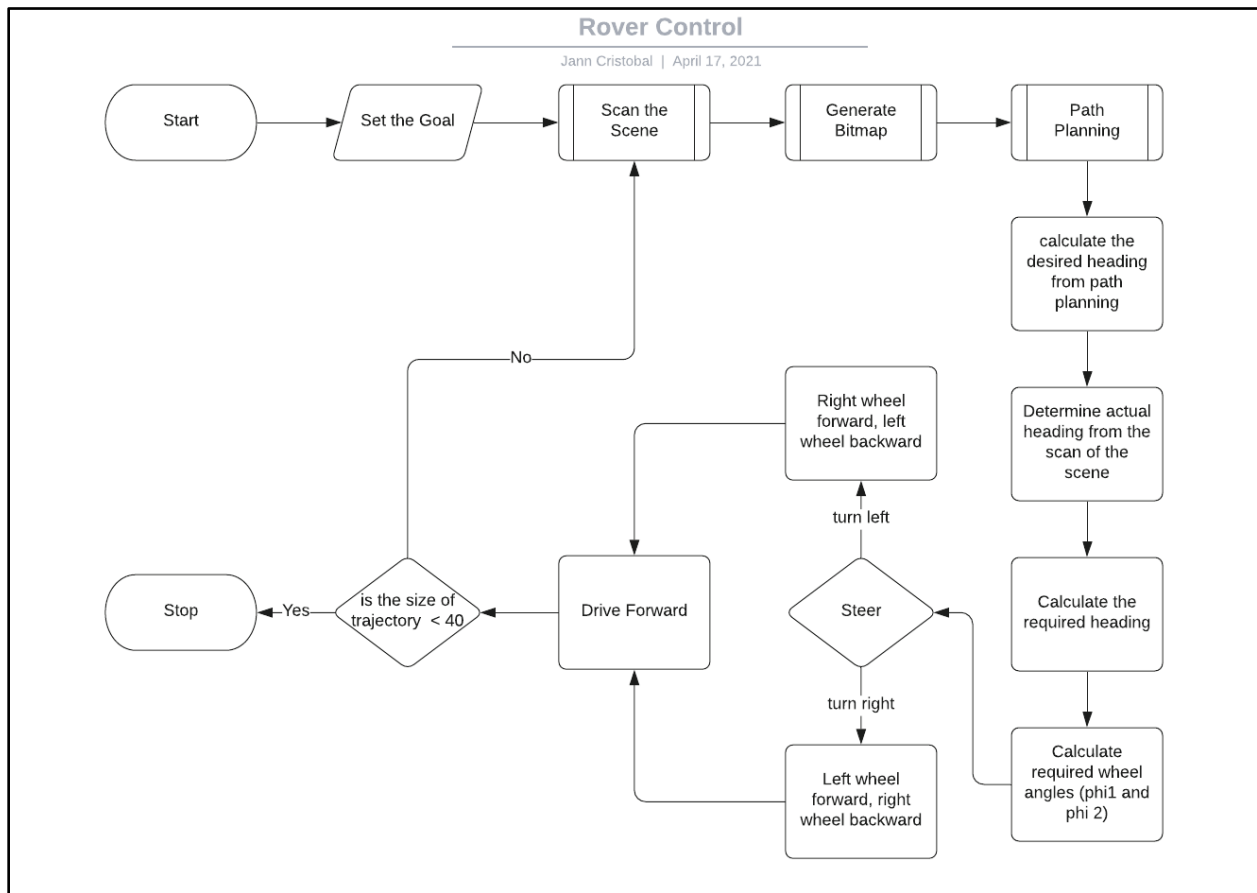
Where  $k_p$ ,  $k_i$  and  $k_d$  are proportional, integral, and derivative gains, respectively. Using a PID controller reduces the steady-state error, improves transient response, and increase stability.

### 3.1.3 Implementation

When implementing the rover control there were two iterations. The first one was generating the path once and trying to follow it throughout. The second one is generating a new trajectory every scan and only following the first 0.5s of it.

The first method proved to be unreliable whenever the vehicle's April tags were not in view of the camera. The moment a failed scan occurred, an error message would show, and the rover would simply drive in a straight line and would stop steering.

The second method proved to be more robust. Whenever there was a failed scan, it would simply use the older data and continue steering at the given time interval until the April tags are in of the camera again.



*Figure 3.3: Rover Control Flowchart*

Figure 3.3 shows the flowchart for how the second method is implemented. The three predefined processes will be discussed in detail in the Package B. The goal is set in the world frame of reference. From the path planning matrix, the desired heading is calculated using trigonometry relating the vehicle's current pose  $(x_1, y_1)$  and location 10 elements forward  $(x_2, y_2)$  as seen in equation 3.1.4.

$$\theta_{desired} = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad (3.1.4)$$

To calculate the actual heading, the transformation matrix of the vehicle is utilized. The average  $\theta_{act}$  was used since the resulting transformation matrix was not a perfect rotation about the  $z - axis$ . Equation 3.1.5 shows the corresponding expression for each cell.

$$T_{0T} = \begin{bmatrix} \cos(\theta_{act}) & -\sin(\theta_{act}) & 0 & x_1 \\ \sin(\theta_{act}) & \cos(\theta_{act}) & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1.5)$$

The required heading is simply the difference between the actual and the desired. For the steering, the rover has to turn right if the required heading is positive, and the has to turn left it is negative. The required wheel rotation is shown in equation 3.1.2

To reach the goal, the size of the trajectory matrix must be less than 40. This was chosen based on the physical size of the rover when compared to the model in the simulation.

### **3.2 Package B: Path Planning (Stephanie Chan)**

Document your path-planning algorithms. Show pictures and corresponding obstacle maps for a variety of simple scenes and plot the generated trajectories on the figures. Explain how your algorithm works (with references as necessary). Highlight the importance of any tuning parameters that you needed to maintain good performance.

## 4 Project Operations

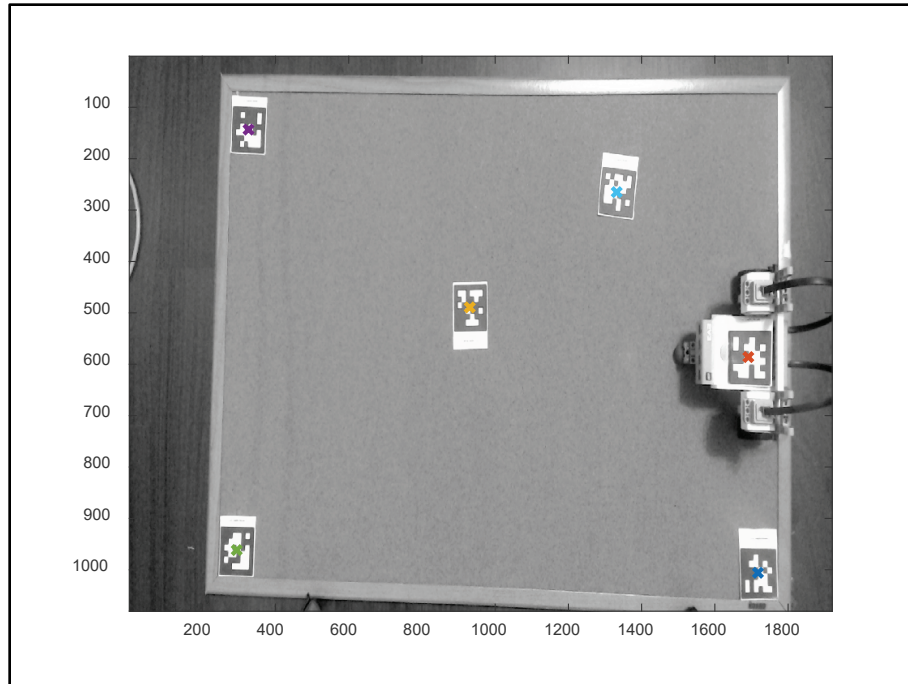
This section answer questions regarding the robot's operation and the results from all the tests when designing and implementing the various parts.

*A. Discuss the integration of the steering controller with the planner. What modifications were necessary to achieve good performance? Did you have to deviate from ideal analysis to get a system that was effective?*

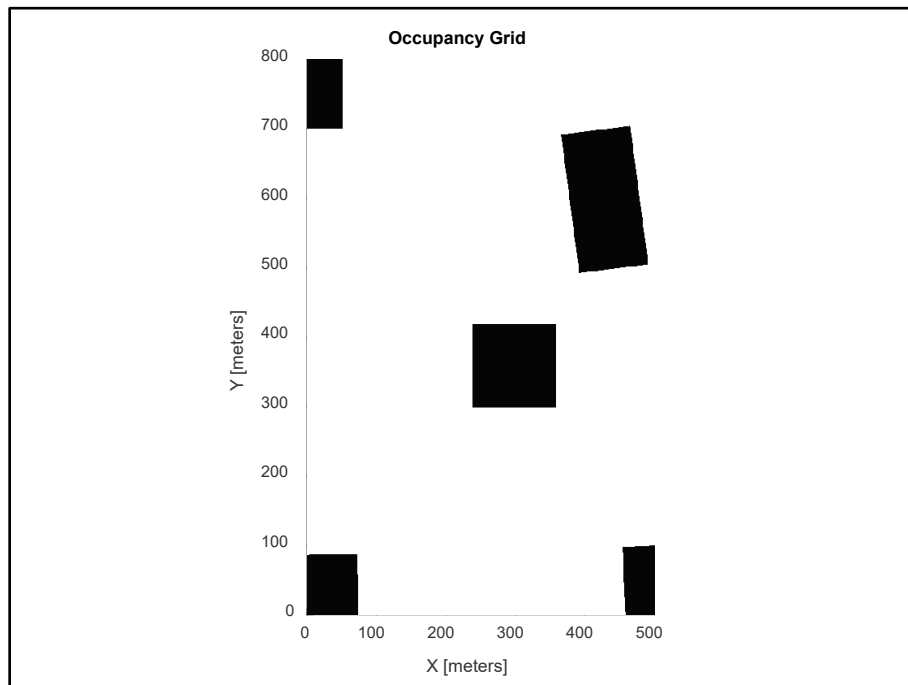
Integrating the steering controller with the path planning algorithm required some modification with the preliminary logic in order to achieve the best performance. Initially, the approach was to generate the one trajectory that will be followed by the rover throughout. The problem with method is that whenever the vehicle April tag was not detected, the whole system would stop running. Measuring the required motor angles also proved to be more difficult and was giving inaccurate motion for the rover. The solution that proved to be more effective was to generate a new trajectory at given time interval and have the robot steer and translate forward for about 0.5 s. The major flaw it had was rotating more than  $20^\circ$  resulted in error propagations and would sometimes not be able to find the goal. The biggest upside this method is it has the ability to continually steer for intervals where the vehicle April tags were not detected. Therefore, the system was more robust. Examples the vehicle April tags not being detected can be seen in the video demonstration. Overall, the final logic resulted in a more effective motion for the rover from start to finish.

*B. Consider the limitations dues to the system hardware. What are the primary limitations of performance? How would you restructure the whole system to get better performance?*

The main limitation in terms of system hardware was the reliability of the April tag detection. Several steps were taken in order to combat this issue particularly recalibrating the camera, and repositioning of the camera to get a top-down view of the scene as seen in figure 4.1. This improved the performance substantially compared when compared to the initial tests where the camera was at an angle, but it did not completely solve the issue. Further steps that can be done to improve the system performance can be redesigning the rover to further minimize the obstruction of the Apriltag, improved scene lighting, larger tag size, higher resolution camera, and manually focusing to the targets. Other minor limitations include inaccuracies with the obstacle pose when placed on a physical obstacle like a small box. This was solved by keeping all the obstacle Apriltags on the flat surface of the scene as seen on figure 4.1. The obstacles can still be seen in the simulation shown in figure 4.2 and 4.3 but are not physically there.



*Figure 4.1: Actual view of the scene*



*Figure 4.2:Occupancy Map of the scene*





## 5 Conclusion

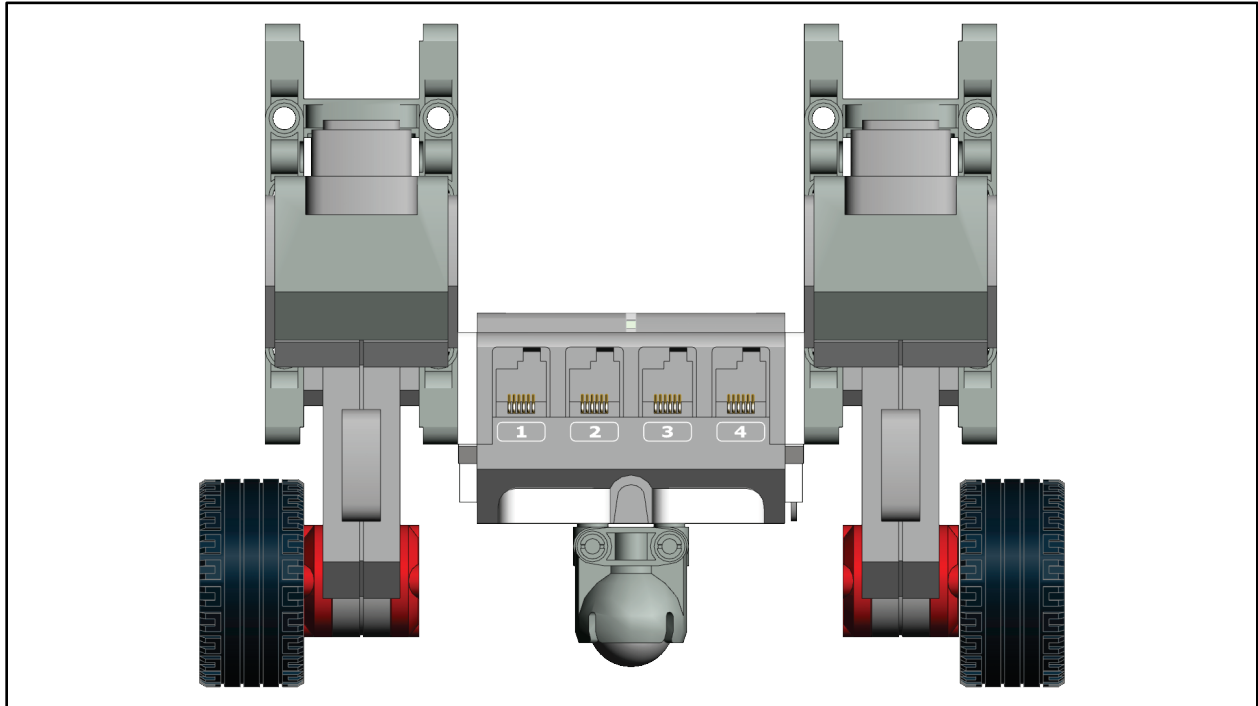
Overall, the system performed as intended. It was able to go from the start position to the goal position for all three test runs. Unfortunately, it was not able to completely avoid the obstacles. This is due to several factors including the large physical size of the rover compared to the model. The path finding algorithm drew path extremely close to the obstacles. The system proved to be robust in situations when the Apriltags on the vehicle were not detected. It continued to navigate towards the goal using older data. The reliability issues with the vision system minimized by repositing the camera and multiple recalibrations. The steering control worked satisfactorily but had issues when targets are in the  $-B_x$  direction.

## 6 References

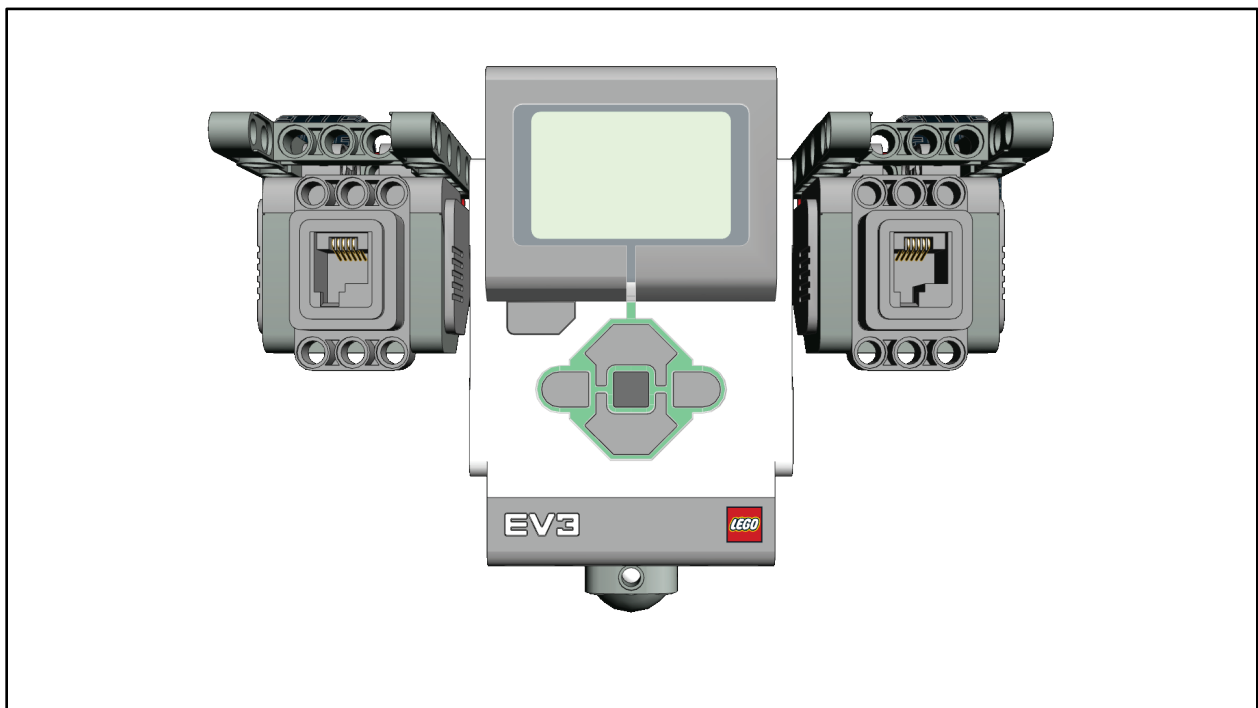
- [1] J. Enright, "AER 627 Project 4," Ryerson University, Toronto, 2021.
- [2] N. S. Nise, "4. Time Response," in *Control Systems Engineering 7th Edition*, Ponomareva, Wiley, 2015, pp. 157-233.

## 7 Appendix

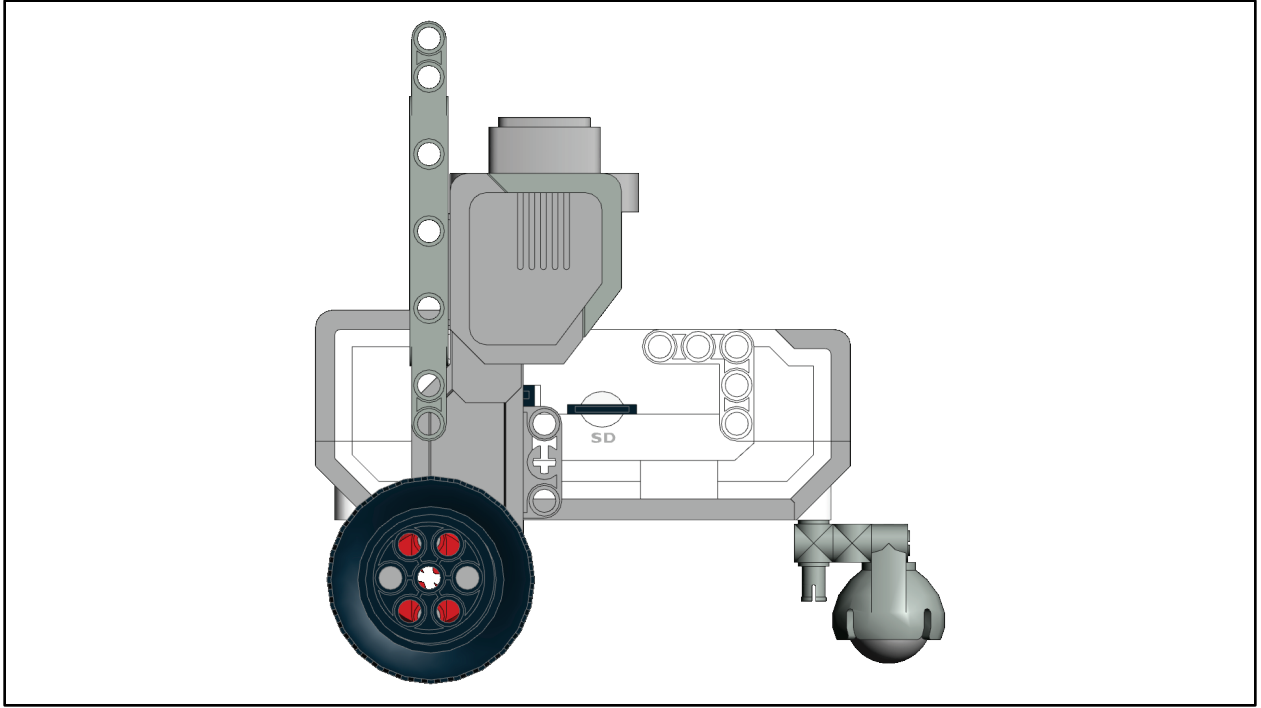
### 7.1 Engineering Drawing



*Figure 7.1: Front view of the rover*



*Figure 7.2: Top view of the rover*



*Figure 7.3: Right view of the Rover*