ENPM662 - Introduction to Robot Modeling

Project 2
Modeling and Simulation of a Martian Rover

SANDIP SHARAN SENTHIL KUMAR
SURIYA SURESH

ABSTRACT

This project explains the design, control, and simulation of a rover with a manipulator in a simulated world. The rover and arm were designed using Solidworks and were exported as a URDF. A ROS package was developed, with transmissions and controllers integrated with the URDF. A camera was attached to the rover to visualize the environment. The kinematics of the robot have been calculated and verified. The rover has been simulated and controlled in Gazebo. A simulation of the rover doing a planned task in a Martian environment has been done. The challenges faced and future work planned have been described.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

Space exploration has always been a topic of fascination since the twentieth century. Recently, many space agencies have launched probes and missions to space to explore it. The rover has been a popular choice to explore planetary surfaces among the machines and robots used. A rover is a vehicle designed to travel across planetary surfaces to collect data. They can be controlled or completely autonomous. Rovers can be used for many purposes, primarily to collect data about a remote environment. They can retrieve, organize and process samples and send the results back. Rovers sent to Mars predominantly focus on gathering evidence of life that might be existed long ago and the feasibility of various sites for future human habituation.

Some of the notable rover missions include the ones sent to space under NASA to Mars, Curiosity, and Perseverance. There have been only five successful landings of rovers on Mars[1]. To date, only NASA and China have successfully landed rovers on the surface of Mars. We took inspiration from NASA's Curiosity and Perseverance Rover for our project.

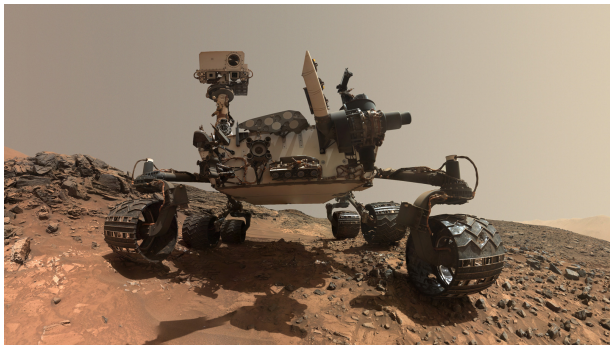

Figure 1.1: Mars Curiosity Rover[2]

This project report describes the work done in designing, controlling, and simulating a rover with a manipulator attached to it. The report is organized under various sections. This section introduces the idea of the rover and what it does. The proposed application and in-depth details specifications of the rover are described under **The Mars Rover**. **Kinematics and Kinematic Validation** describes the forward and inverse kinematics of the manipulator of the rover and verification of

theoretical calculation with various poses of the arm of the rover using Peter Corke's Robotics toolbox. The **Workspace Study** explains the total reach of the arm and the limitations of the space it cannot reach. The various assumptions made and taken into account whilst developing the rover and for designing a method of control for the rover are described under **Assumptions**. The rover is simulated in an empty world and a designed world with a developed control logic that is described in **Control Method and Visualization**. Along the journey of developing this model, we have faced many issues and the points that we took from them to learn and progress further are explained in **Problems Faced and Lessons Learnt**. There has been a lot of work done but there are certain components that can be always improved. Those improvements have been listed under **Future Work**. The **Conclusion** of the project summarizes the overall project. The **References** section lists the references used for the project.

1.1 Goals of the Project

The goals laid for this project are:

- Design a Martian Rover and a 5 DoF manipulator.
- Calculate and verify the kinematics of the manipulator.
- Simulate the rover in a Martian Environment.
- Control the rover in simulation.
- Execute the task of drilling into the surface in simulation.
- Integration of a camera onto the rover.

Fallback Goals

- Designing a similar rover with 4 wheels instead of 6 wheels by removing rocker-bogey suspension.
- Designing a robotic arm with lesser degrees of freedom.
- Changing the drill manipulator to a pick and place manipulator.
- Removing the microscopic camera present in the rover and replacing it with a regular camera

- Replacing the Martian environment developed in Gazebo with a simpler rough terrain which is similar to Martian terrain.

Ambitious Goals

- Modifiable manipulator - The end effector is modifiable and can be changed autonomously based on the requirements.
- Suction drill - The drill end effector uses suction to collect the soil from the drilled area in the terrain.
- Microphones - To collect the local audio of the environment.
- Thermal camera - Inclusion of thermal camera in gazebo environment to sense the thermal composition of Martian terrain.
- Ground Penetrating Radar - This is used to map the geographical features under of simulated gazebo environment.

1.2 Learning Outcomes

The learning outcomes of the project that we aim to achieve:

- To understand design and modeling software such as AutoCAD, Solid works, and ANSYS
- Understanding how the axis and the coordinate systems for the robot can be defined and the model can be exported as a URDF file.
- To understand the parent-child link of the URDF exporter in Solid works.
- To learn about the tools like Gazebo & RViz.
- To understand & calculate the forward and inverse kinematics for the links in the rover's robotic arm.
- To integrate sensors and cameras into the robot model and simulate them in the gazebo environment.
- To validate the calculated forward and inverse kinematics of the rover using simulation.
- To understand and implement the controllers required to navigate the rover in the simulated environment.

Chapter 2

THE MARS ROVER

This section explains the proposed design[3] [4], specifications, and the planned task of the Rover being modeled for this project.

2.1 Specifications of the Rover

Rover	
Length of the Rover	714 mm
Width of the Rover Body	289 mm
Height of the Rover without Arm	331 mm
Mass	12022g
Wheel Track	604 mm
Wheel Base	710 mm
Power	6 individual motors to all 6 wheels
Suspension	Rocker-Bogie Mechanism
Degrees of Freedom	3 in total.Along x and y directions and yaw direction of rotation.
Rover Arm	
Height of Arm with all joints extended	655 mm
End Effector Used	A rotary percussive drill
Degrees of Freedom	3 in total. Two Revolute and one Prismatic. Additional 1 DoF if considering the rotation of end effector
Length of End Effector	169 mm

2.2 Type of Robot

The type of robot used here is a rover, which can be considered a space exploration robotic vehicle that can be used to explore the surface of a planet[5]. A rover can perform many operations autonomously. It can also be controlled manually to do tasks. They can be used as transport vehicles for humans as in the Moon Rover or for research as in the case of Mars Rovers currently.

2.3 Degrees of Freedom of The System

The system has a total of 7 DoF. The rover has 3 DoF while the arm has 4 DoF. The rover can move along x and y directions and rotate along the yaw axis. The arm has 2 revolute joints, a third prismatic joint, and an end-effector rotation.

2.4 Application of The Rover

The end effector of the Rover has a drill attached to it that can rotate at high speed. We plan to use this rover to drill into the surface of a simulated Martian Environment[6]. Using the drill, the rover will be able to collect soil samples from underneath the surface of the environment.

2.5 Motivation for Choosing the Mars Rover

The area of space exploration using rovers intrigued us and the opportunities that this field offers fascinated us to take upon this topic.

This project presented an opportunity to model a real-world rover in simulation and get experience with the mechanics involved in positioning and controlling the robot. The challenges that would be faced in trying to control a rover would give the experience that could be applied to similar situations in real-time applications.

Chapter 3

CAD MODELS

This section contains pictures of the rover assembly and the arm assembly from various perspectives.

3.1 The Rover Model

The model of the rover was sourced from Github[3] and was mated with an arm, similarly sourced from Github[4]. The model of the rover is based on the publicly available NASA-provided models of their Mars Rovers[7].

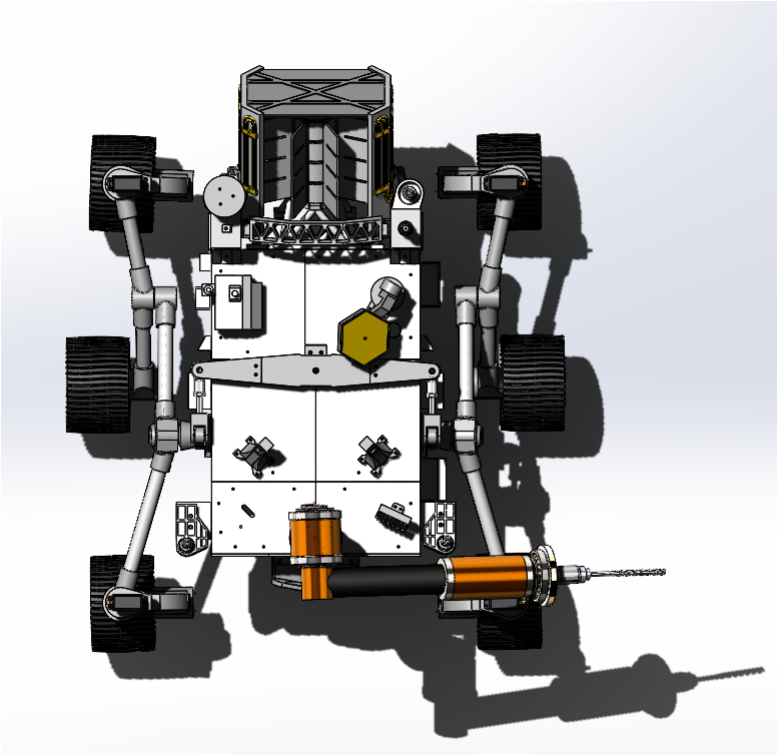

Figure 3.1: Top View of the Rover

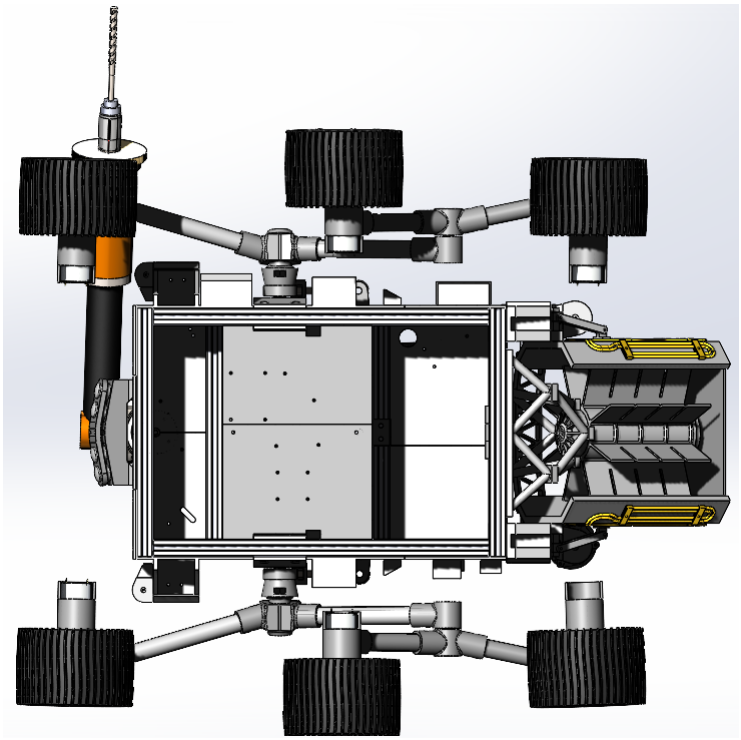

Figure 3.2: Bottom View of the Rover

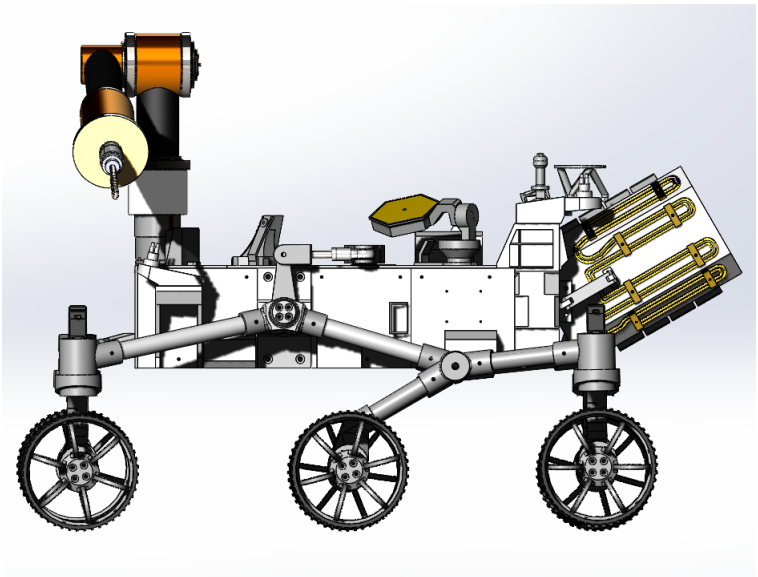

Figure 3.3: Right Side View of the Rover

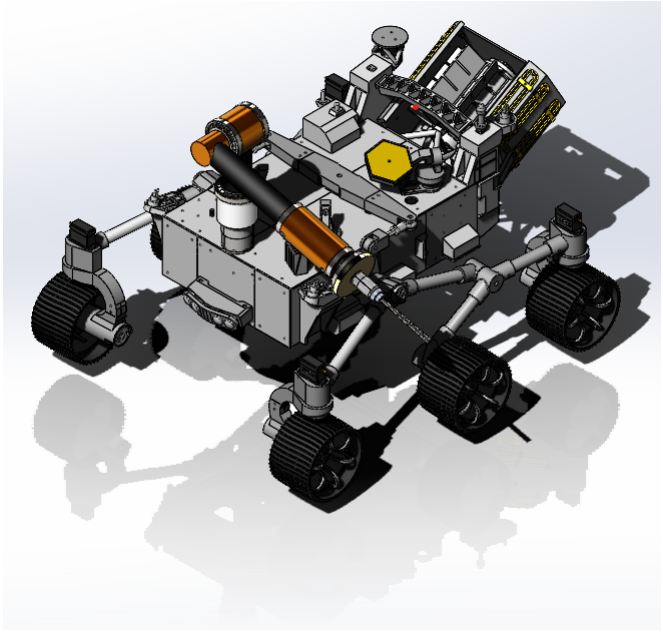

Figure 3.4: Top Right View of the Rover

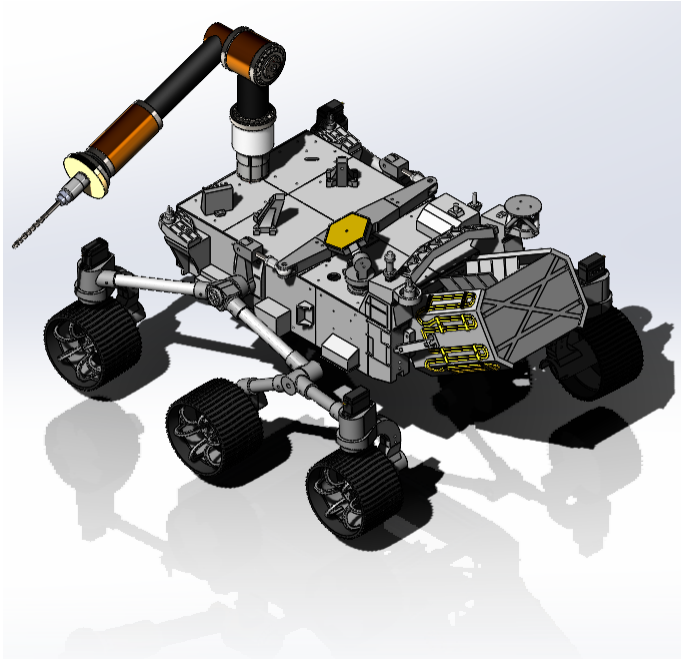

Figure 3.5: Rear Right View of the Rover

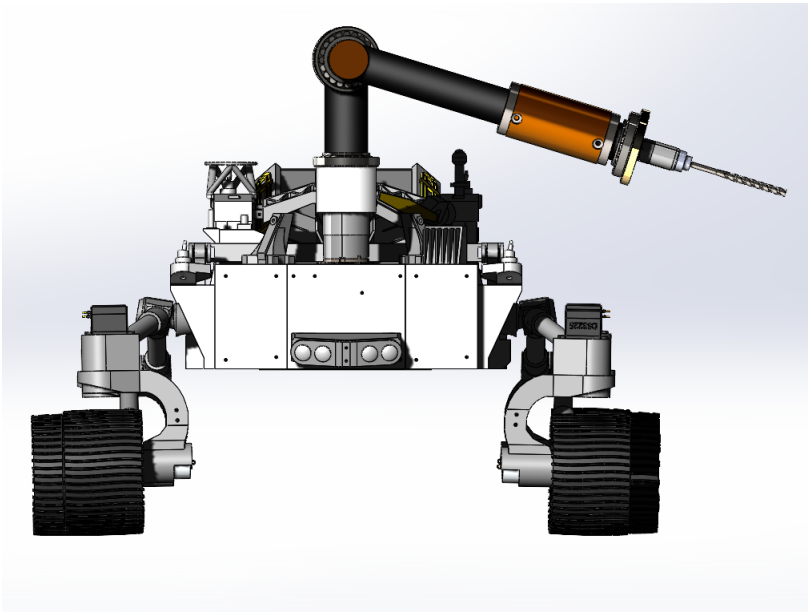

Figure 3.6: Front View of the Rover

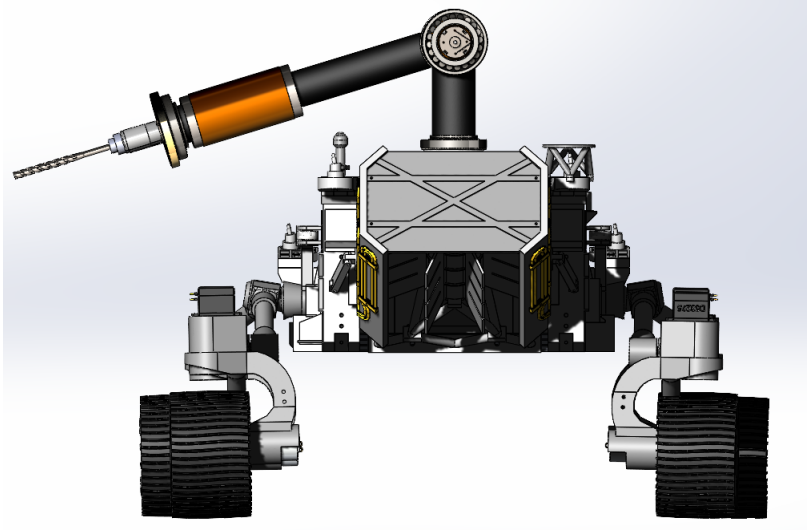

Figure 3.7: Rear View of the Rover

3.2 The Arm Model

This section has various views of the arm assembly. This has 4 joints, the base, one revolute joint in the middle of the arm, a prismatic joint attached to the end effector, and the end effector can rotate.

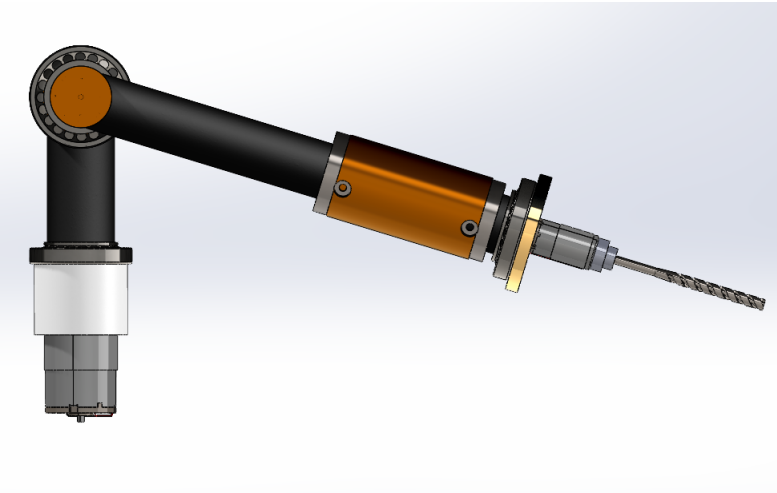

Figure 3.8: Side View of Arm

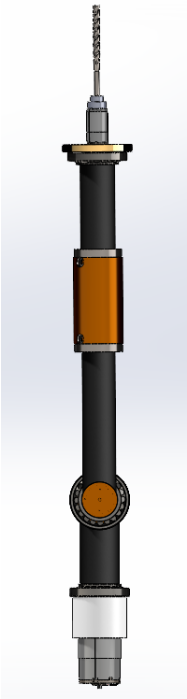

Figure 3.9: Fully Extended Front View of Arm

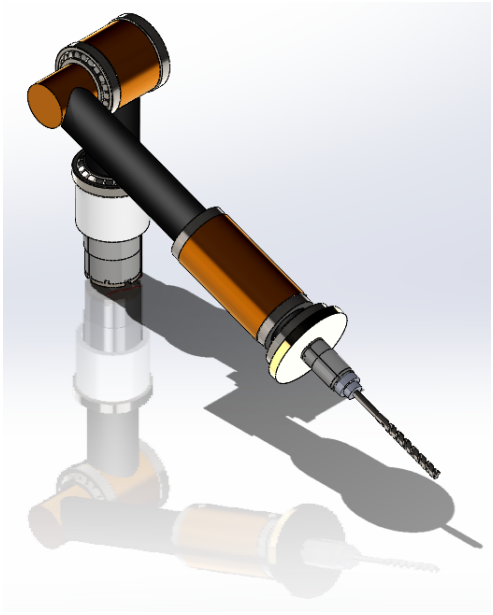

Figure 3.10: Top Right View of Arm

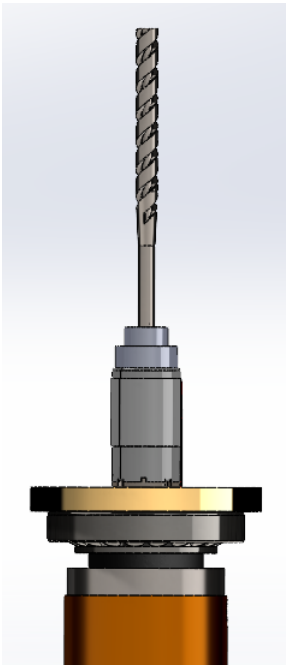

Figure 3.11: Close-Up of End Effector

Chapter 4

KINEMATICS

4.1 Forward Kinematics

The Forward Kinematics part is calculated only for the manipulator ignoring the rover body. Calculating the forward kinematics can be done by following the steps below:

- Assigning the frames at each joint
- Finding the D-H table
- Calculating the Final Transformation Matrix

For Assigning the frames, the manipulator we have chosen is a 4 DoF Robotic manipulator which has 2 revolute joints and a prismatic joint and has a rotating end-effector with a drill tool. As we have already mentioned that the manipulator has a total of 3 joints which means we have to assign 4 frames including the end-effector frame.

The base frame has been assigned at the base of the manipulator 1^{st} revolute joint which is attached to the rover body. The second frame has been assigned at the 2^{nd} revolute joint. The third frame has been assigned at the prismatic joint and the final frame is located at the tip of the end-effector. The frame assignment is shown in the image below. Shifting the origin method is used to assign frames so that it adheres to the rule where the X_n axis is perpendicular and intersecting the Z_{n-1} axis.

The D-H table is calculated by using the assigned frames and the calculated D-H table is shown in Table 1. The D-H table is calculated by following the rules:

- θ_n is the orientation given by the revolute joints
- q_n is the linear movement given by the prismatic joint c
- a_n is the distance between the Z_{n-1} and Z_n (along X_n axis)
- d_n is the distance between the O_0 and intersection of Z_n and X_n along Z axis
- α_n is the angle between Z_{n-1} and Z_n

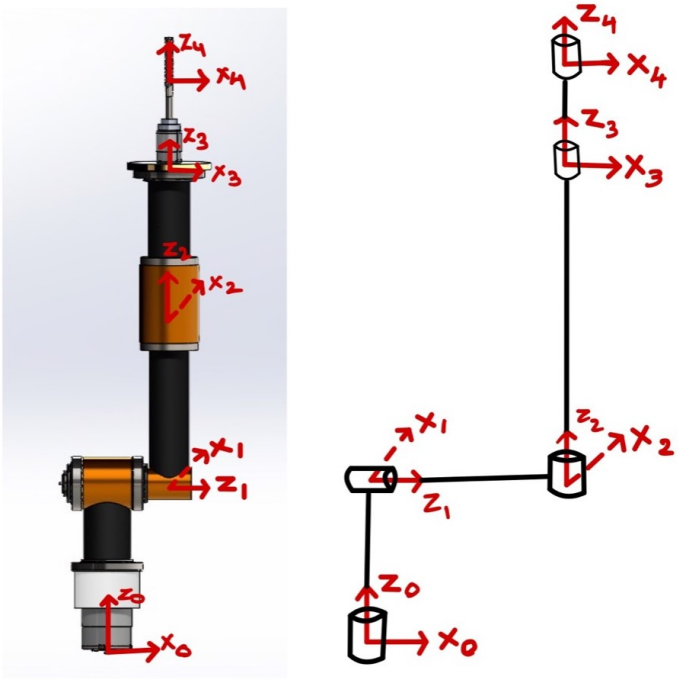

Figure 4.1: Assigned frames in the Robotic Arm

D-H TABLE				
	α	θ	d	a
0 - 1	$-\pi/2$	$\theta_1 - \pi/2$	d_1	0
1 - 2	$\pi/2$	θ_2	0	a_2
2 - 3	$-\pi/2$	0	q_3	0
3 - 4	0	θ_4	0	0

Then using the D-H table and the assigned frames we calculated the final transformation matrix which gives the pose and orientation of the end-effector. To find the final transformation matrix we have to follow the following steps:

1. Individual Transformation matrices are to be found (i.e.) T_0^1, T_1^2, T_2^3 and T_3^4
2. The individual transformation can be found by plugging in the joint angle values and the rest of the values from the D-H table given above into the below matrix

$$\begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. The final transformation can be found by multiplying all the individual transformation matrices in the format given below

$$T_0^4 = T_0^1 * T_1^2 * T_2^3 * T_3^4$$

4. The final column of the final transformation gives us the pose of the end-effector in X, Y, and Z coordinates where

   P_x = 1^{st} element of that column

   P_y = 2^{nd} element of that column

   P_z = 3^{rd} element of that column

5. The first 3 rows and columns give the orientation of the robotic arm

The Transformation matrices are given below:

$$T_1 = \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & 0.222 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0.6553 \cdot \cos(\theta_2) \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 0.6553 \cdot \sin(\theta_2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.2: Transformation Matrices 1 and 2

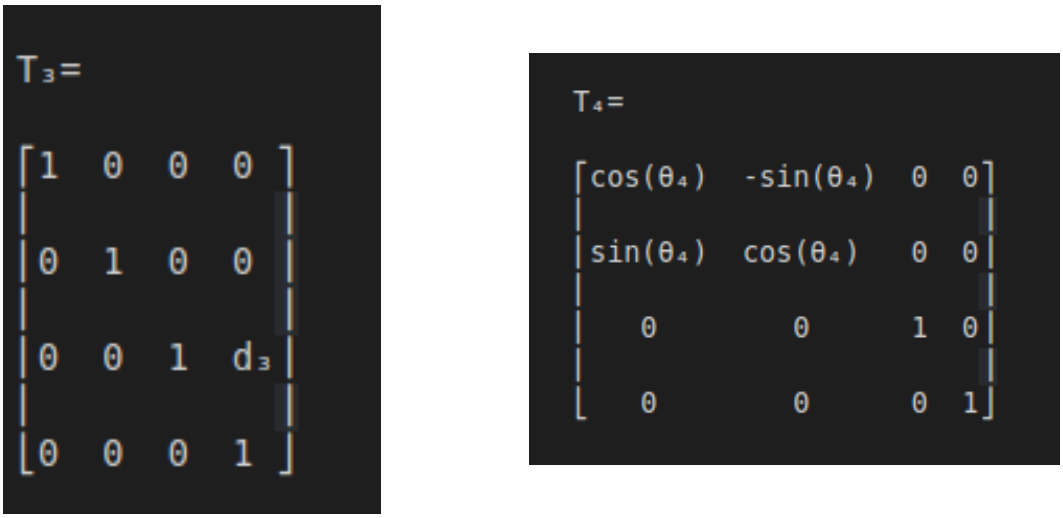

Figure 4.3: Transformation Matrices 3 and 4

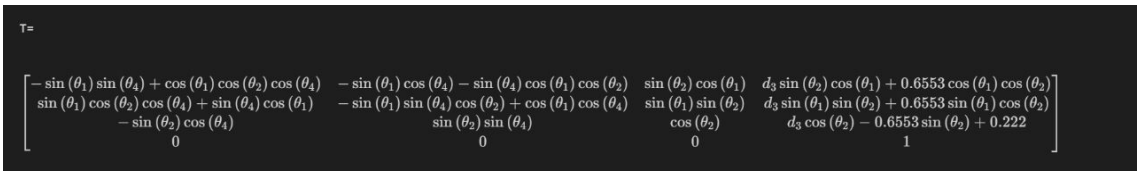

Figure 4.4: Final Transformation Matrix

4.2 Inverse Kinematics

In forward kinematics, we calculated the end-effector position using the given joint angles. In inverse kinematics, we will go in the opposite direction by calculating the joint angle using the known end-effector position. There are two different methods to calculate inverse kinematics

- Inverse position kinematics
- Inverse Velocity Kinematics

We have followed the path of inverse velocity kinematics. To find the joint angles by using the end-effector position we have to follow certain steps. They are:

1. Finding all the individual Transformation matrices and the final transformation by multiplying them
2. Find the Z_i matrices are the column matrices containing the elements of the 3^{rd} column in the T_i matrix

3. After finding the Z_i matrices, Then find the X_{pi} matrices which contain the elements from the last column of each individual matrices

4. Differentiate the matrices X_{p1}, X_{p2}, X_{p3} and X_{p4} with respect to the joint angles θ_1, θ_2, q_3 and θ_4 respectively

5. Then form the individual jacobian matrices in the format given below

$$ {}^0J_n = \begin{bmatrix} \frac{\partial X_p^0}{\partial q_1} & \cdots & \frac{\partial X_p^0}{\partial q_i} & \cdots & \frac{\partial X_p^0}{\partial q_n} \\ {}^0Z_1 & \cdots & {}^0Z_i & \cdots & {}^0Z_n \end{bmatrix} $$

6. Since we have a prismatic joint at joint 3 the ${}^0Z_3$ matrix becomes 0

7. After finding the final jacobian matrix, the inverse of the respective Jacobian matrix should be found. In this case, it is a 4 x 6 matrix so the matrix will not be invertible so we have to find the pseudo-inverse of the matrix.

8. Then find the $\dot{Q}$ matrix by multiplying the inverse jacobian matrix and the $\dot{X}$ matrix

   where ($\dot{X}$ matrix is the matrix that contains the linear and angular velocity components)

9. The $\dot{Q}$ matrix contains the values of the joint angles and those values are obtained and numerical integration is performed to get the corresponding joint angle values

10. Then finally, the final joint angle values are calculated using the final transformation matrix at the final iteration

J=

$$ \begin{bmatrix} -q_3 \sin(\theta_1)\sin(\theta_2) - 0.6553\sin(\theta_1)\cos(\theta_2) & q_3\cos(\theta_1)\cos(\theta_2) - 0.6553\sin(\theta_2)\cos(\theta_1) & \sin(\theta_2)\cos(\theta_1) & 0 \\ q_3\sin(\theta_2)\cos(\theta_1) + 0.6553\cos(\theta_1)\cos(\theta_2) & q_3\sin(\theta_1)\cos(\theta_2) - 0.6553\sin(\theta_1)\sin(\theta_2) & \sin(\theta_1)\sin(\theta_2) & 0 \\ 0 & -q_3\sin(\theta_2) - 0.6553\cos(\theta_2) & \cos(\theta_2) & 0 \\ -\sin(\theta_1) & \sin(\theta_2)\cos(\theta_1) & 0 & \sin(\theta_2)\cos(\theta_1) \\ \cos(\theta_1) & \sin(\theta_1)\sin(\theta_2) & 0 & \sin(\theta_1)\sin(\theta_2) \\ 0 & \cos(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix} $$

Figure 4.5: Jacobian Matrix

Chapter 5

KINEMATIC VALIDATIONS

Both the forward kinematics validation and the inverse kinematics is done using Peter Corke's Robotics Toolbox. We used the joint angle values and the end-effector position to validate the kinematics of the robotic arm.

5.1 Forward Kinematics

For validating the forward kinematics of the Robotic arm, we fed the D-H table parameters and modeled the Robotic arm using Peter Corke's Robotics toolbox in MATLAB. Then gave all the joint angle values "0 (zero)" so that it is in the home position(case (1)) and the X, Y, and Z values printed in the figure matched with the translation part of the final transformation matrix which is shown in figure 4.1. Then the value of the prismatic joint is given the maximum value in this case it is "1 (one)" and figure 4.2 shows the validation for case (2) - extended prismatic joint position of the Robotic Arm. The Transformation matrices at the same position of the robotic arm are also shown to the right respectively. Using these transformation matrices we can validate the forward kinematics.

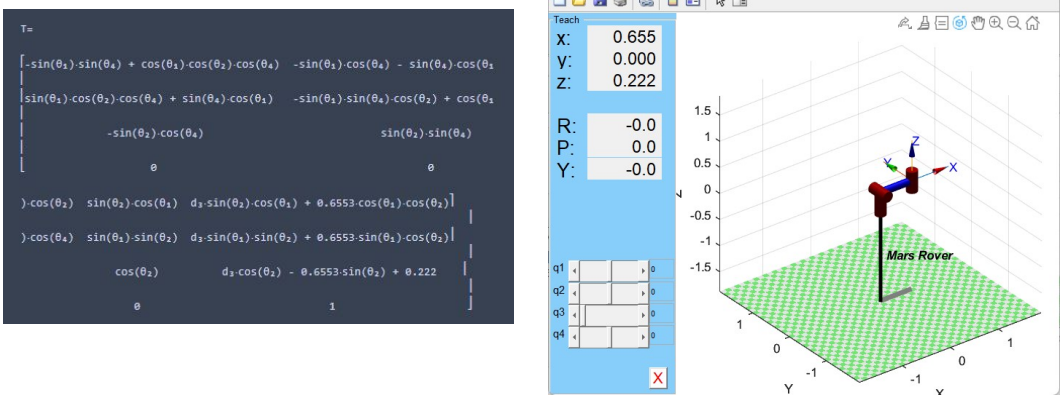

Figure 5.1: Home Position Validation for the Robotic Arm

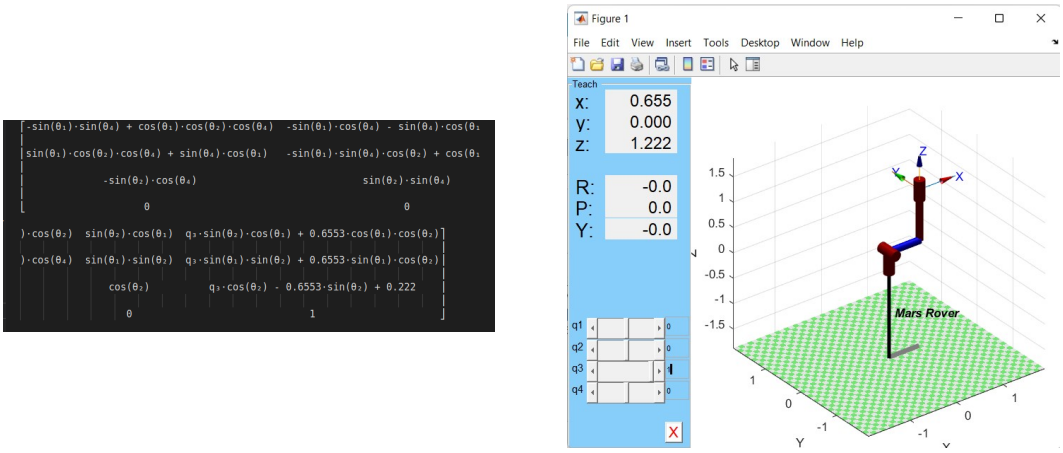

Figure 5.2: Extended Prismatic Joint Position Validation for the Robotic Arm

5.2 Inverse Kinematics

For validating the inverse kinematics part, we validated it using two different methods

1. Plotting a circle using inverse velocity kinematics
2. Providing the joint angles obtained at a random end-effector position as an input so that the translation part matches with the X, Y, and Z values in the figure

1st method: We wrote a python script that plots a circle by finding the jacobian matrices and doing numerical integration for all the joint angle values and plotting the joint angle values obtained in each iteration.

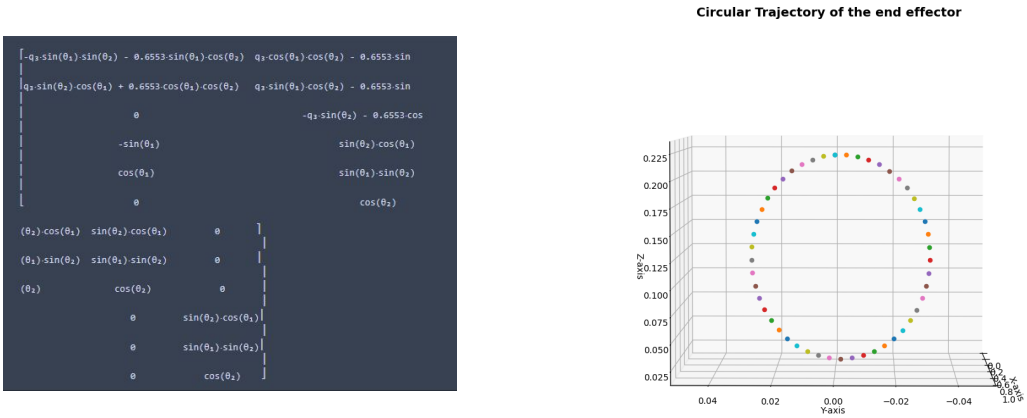

Figure 5.3: Jacobian Matrix and the Circle Trajectory

Chapter 6

WORKSPACE STUDY

We did the workspace study of the rover arm assuming it is not attached to the rover. It forms a sphere as seen in Fig 5.1. This sphere is achieved when the prismatic joint is extended to its maximum and all joints rotate 360 degrees.

The prismatic joint can extend a max length of 250 mm from the previous joint. The revolute joints have no constraints unless they are attached to the rover. The arm does not have singularity and can reach all points in its workspace. As seen in fig 5.1, the base can rotate 360 degrees.

When the arm is attached to the rover, the base joint can rotate 360 degrees. The second joint is restricted to a range of angles from 0 to almost 300 degrees as the arm collides with the body. The prismatic joint has no restrictions in extending apart from physical constraints.

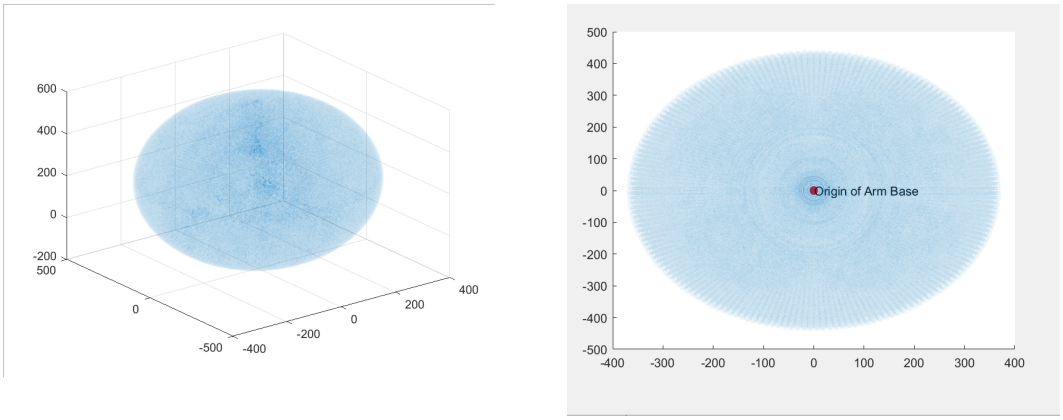

Figure 6.1: Workspace of the Robotic Arm

Chapter 7

ASSUMPTIONS

We did a few assumptions which allowed the rover to be modeled without getting too complicated. The assumptions involve assuming the properties of the rover, environment, forces acting on the system, and the manipulator. The assumptions done are:

- The model is a scaled-down of the original.
- All links of the rover are considered to be rigid.
- The wheels have cleats for traction.
- External forces like friction have not been considered.
- The planned path of the rover and arm is one of the many possibilities for the certain path and tasks planned.
- There are fewer sensors as compared to the original rover.
- The rover's rocker-bogie suspension has not been given any commands. It is allowed to act freely within limits.
- The communication mast and a few other parts have been removed from the model as they affect the center of gravity of the rover and collide with the arm when it rotated.

Chapter 8

CONTROL METHOD AND VISUALIZATION

8.1 Control Method

The rover contains 6 wheels connected using the Rocker-Bogie mechanism and has a robotic arm attached in front. The robotic arm attached has 4 joints (3 revolute joints and 1 prismatic joint). So, in total, 18 joints are to be controlled(6- wheels, 4-steer, 4- arms, 4- rocker-bogie)

- For wheels we have selected velocity controllers since they have to keep rotating at a continuous velocity till it reaches the expected position
- For the rocker-bogie mechanism we need a precise angle to control so we selected Joint Position controllers for this reason
- For controlling the robotic arm we have selected Joint Position controllers because of the reason mentioned above

The 18 joints which require the controller are shown in Figure 8.1 with their respective controller and their PID values.

These controllers internally use a PID controller for precise movement of the joints. Each joint has a specific range limit, hence any input to the controller in that range will rotate the joint to the desired location. Then the PID values should be tuned manually to avoid the jitters in the joints. The tuning can be done with the help of the rqt tool from the ROS and tuned accordingly to minimize or completely eliminate the jittering. Then these tuned PID values are passed on to the controller before spawning in the environment.

The simulation video can be watched using the drive link given below:

SIMULATION VIDEO

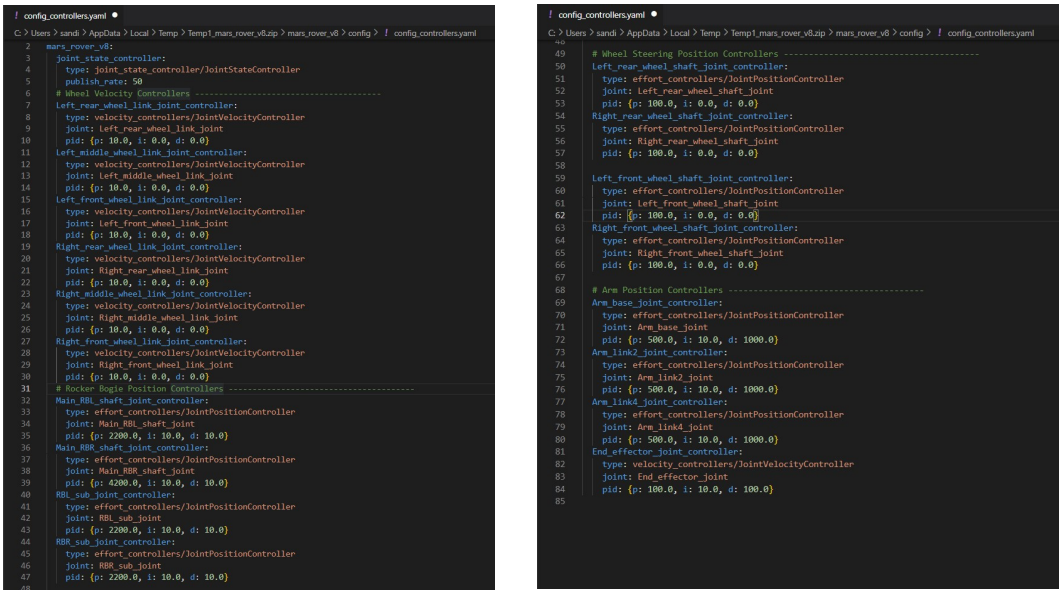

Figure 8.1: Joints and their controllers with their tuned PID Values

8.2 Simulation

The simulation for this project was done using Gazebo and RViz. For simulating the rover the steps followed are given below:

- After building all the required files in the ROS package, launch the rover using the template launch file.
- The rover can be seen spawning in the Martian world but due to heavy render the real-time factor drops down to 0.01 which slows down the rover.
- Then open the RViz and load the robot model in the software to visualize the view of the rover's camera, add the camera option in RViz
- Then move the rover around by passing the command using a python script

Figure 8.2 and Figure 8.3 shows the rover being spawned in the Martian World[6] and Figure 8.4 shows the rover being spawned in the Empty world. The simulation videos are attached in the link given below. Figure 8.5 shows the rover being visualized using RViz software and the camera view is also shown on the right bottom of the image

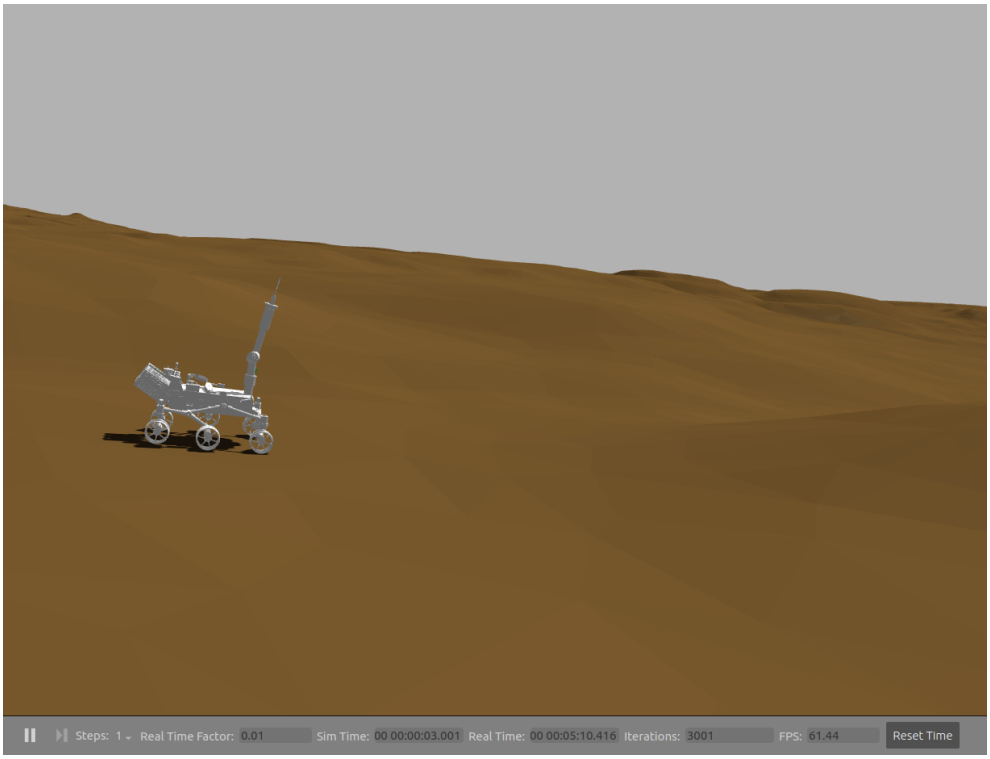

Figure 8.2: Rover spawned in Martian World (Side view)

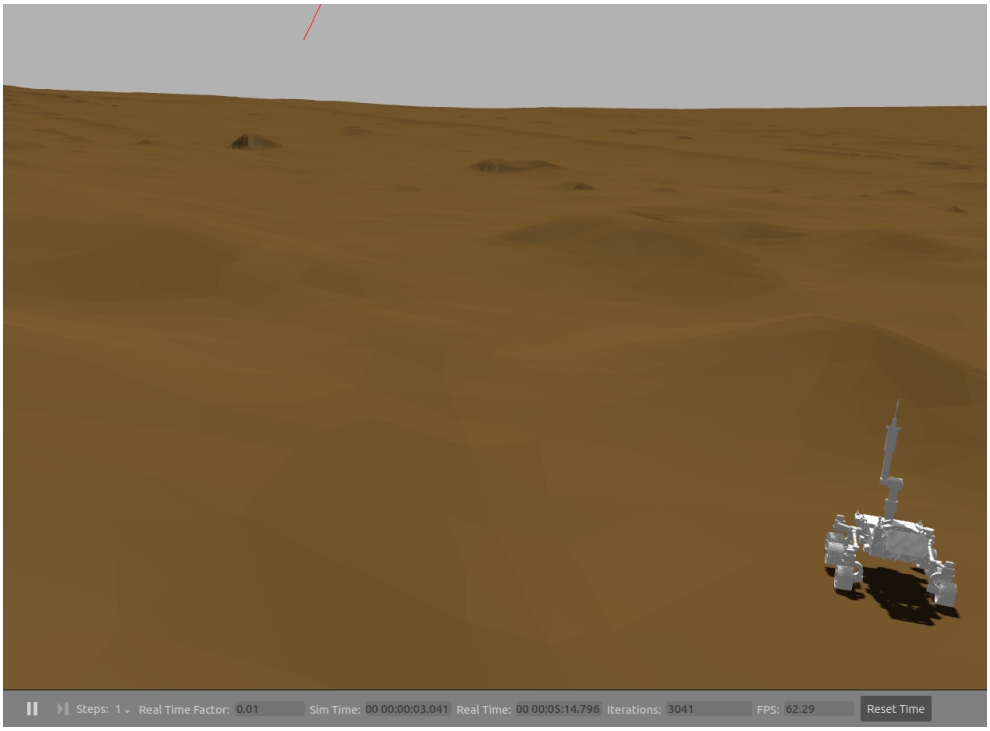

Figure 8.3: Rover spawned in Martian World (Rear view)

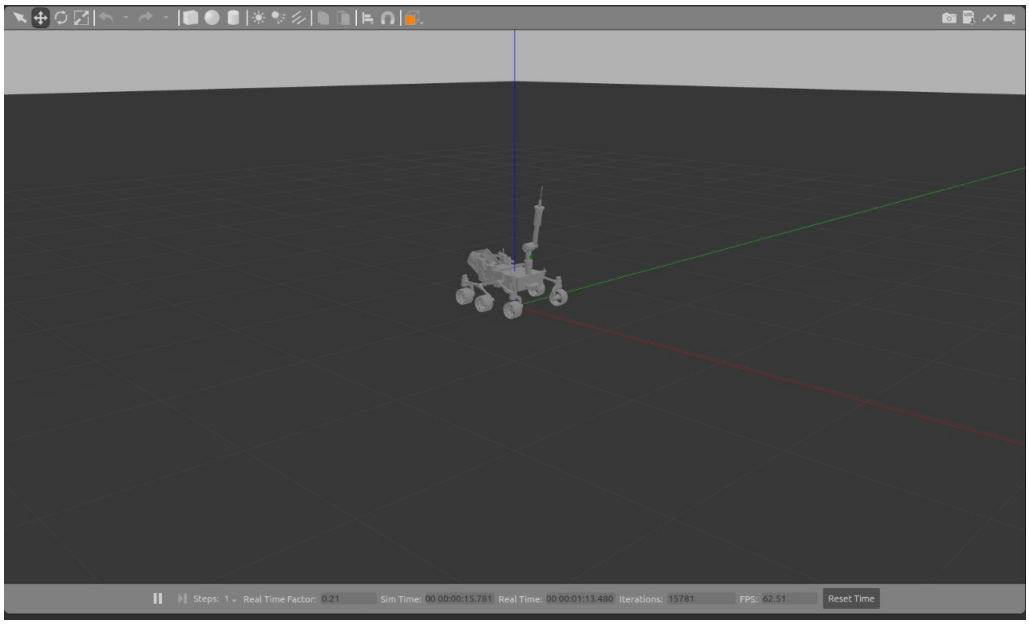

Figure 8.4: Empty World Simulation

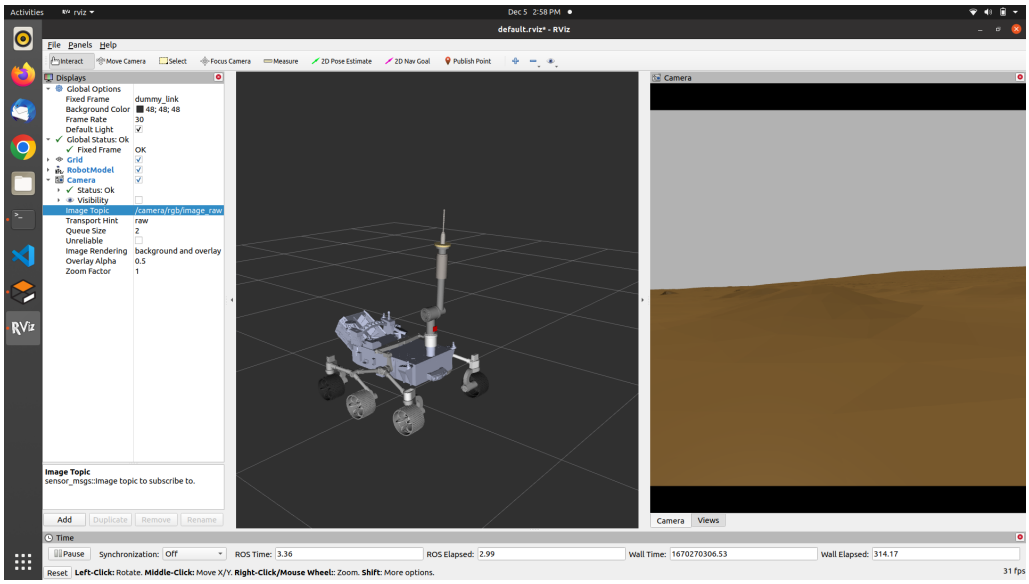

Figure 8.5: RViz Simulation

Chapter 9

PROBLEMS FACED AND LESSONS LEARNED

9.1 Problems Faced

This section describes the issues faced during the progress of the project. They have been described in depth and the solutions implemented to overcome them have also been listed.

Improperly Defined Axis in URDF

Exporting the URDF was done for many iterations. For every iteration of exporting the URDF, it was rendered in Gazebo where it fell apart. The issue was the axis definitions being a bit askew in the model in Solid works. Proper axis definitions along with appropriate coordinate systems resolved the issue.

Choosing appropriate controllers for the Suspension

The rocker-bogie mechanism enables the rover to have great maneuverability in rough terrain. The initial controller that was given was a joint position controller which affected the movement of the arms of the suspension. The controller was switched to effort controllers which made the suspension work as expected.

Reach of Arm

In the final stages of simulation, when the rover was being tested for performance of the planned task of drilling into the surface of the world, the arm when fully extended, the end effector did not touch the ground. The arm had to be re-positioned in the CAD model and the pipeline of creating the ROS package had to be done again from the start. The arm was shifted forward closer to the edge of the rover while being along the middle of the width of the rover.

Positioning and Visualizing the Camera

The camera is positioned at the front of the rover. Its positioning had to be readjusted multiple times as the view was obstructed by the body of the rover and adjustments had to be done to ensure it did not slant too much toward the ground. Visualizing the camera in RViz had a top-down perspective view which was confusing, the fix was deselecting visualize in RViz.

Jittering of Wheels in Gazebo

The wheels of the rover were jittering badly after the addition of controllers to the ROS package. Assumed PID values were causing the wheel shaft attached to the wheels to vibrate badly after spawning the model in the Gazebo world. This resulted in the model not being stationary when no command was given and the rover was at the rest. This motion resulted in the model going in random directions. Tuning the PID values using the **rqt** offered by ROS allowed for a live view of how the wheel reacted to changing the Proportional Integral and Derivative components of the controllers. Tuning values decreased the vibration of the wheels and resulted in the expected behavior of wheels being stationary when no command is given to the wheels.

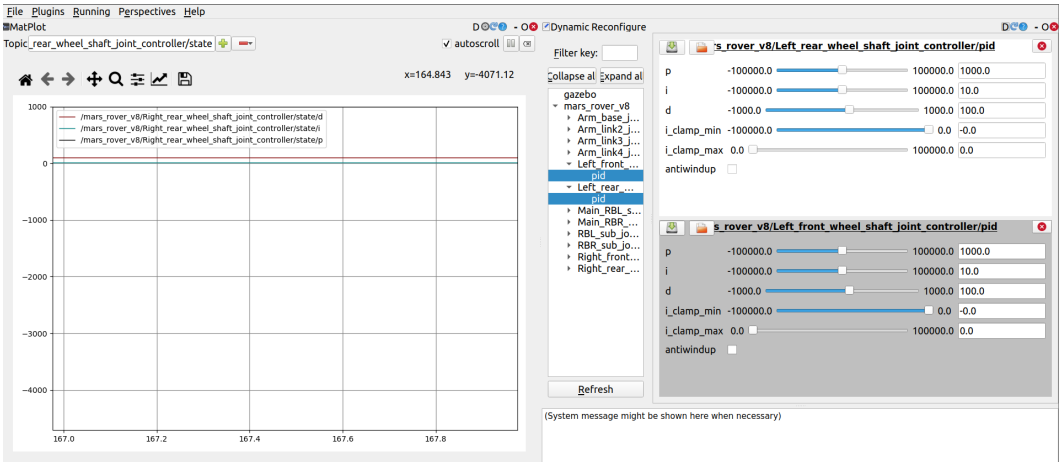

Figure 9.1: Using RQT to tune PID values

Low Real-time Factor in Martian World in Gazebo

Importing the Martian environment caused the real-time factor to dip to 0.01 which slowed the simulation a lot. We had to simulate our task in an empty world as the task was executed very slowly in the martian environment.

9.2 Lessons Learnt

The work done for this project posed a learning curve for us. We have learned a lot from doing this project and continue to progress further as we improve the ROS package and add more functionality.

This project gave us a deeper insight into the work and the pipeline for developing a ROS project from scratch for modeling a real-time application of a robot. We

learned many tips and tricks while exporting a URDF of a model from Solid works. We also learned about editing the URDF file directly through a series of trials as the mass, friction, and damping values were modified as the model was not stable when spawning in Gazebo.

The use of a custom world in gazebo taught us about the various 3-D file formats that Gazebo and ROS support such *.dae* and *.stl*. We also learned about mesh-based 3D file formats that are not easy to export as a URDF compared to doing it via Solid works or Fusion360. We also learned the nuances of importing a model into a gazebo as that posed some challenges as the gazebo refused to load the custom world without sourcing the path of the directory of the model in *.bashrc*. We also learned the use of Move-it, a powerful tool for path planning and executing them. This aided in executing the planned task of drilling a hole into the surface of the environment.

The forward and inverse kinematics part was a major part where we learned a lot. Doing the DH table for a custom model whilst trying to validate the kinematics, taught us about assigning frames, shifting origins, and dummy frames properly as small mistakes would lead to the theoretical values and simulated values not matching with each other. Choosing controllers taught us a lesson in doing improper guesses for assigning appropriate controllers to each joint as it caused the model to react in unexpected ways when spawning in Gazebo and similarly when giving commands to joints.

Chapter 10

FUTURE WORK

This project has been done successfully. There is always a scope for improvement. Some of the improvements we do think of relate to adding more functionality and smoothing out minor issues. Some of the work we plan on doing in the future are:

- Adding a LIDAR to the rover for obstacle detection
- Increase the Degree of Freedom of the arm so that it can execute more range of motion for various tasks.
- Adding a rotating assembly with multiple tools as the end effector for doing different tasks.
- Adding more cameras around the rover for more views of the environment the rover is in.
- Add more sensors to the rover to collect more information about the environment it is in.
- Implementing autonomous navigation for the rover from a start point to a target point.
- Adding a feedback mechanism for the rover to correct the position of the arm when executing tasks if a position error is present.
- Introduce the rover into a more detailed and complex environment and analyze the behavior of the wheels and their mechanism of it.
- lacing a suction drill instead of a regular drill to increase the efficiency of the rover

Chapter 11

CONCLUSION

The project has been completed. The study of a rover in a virtual environment was done successfully. The design, simulation, and control of a rover were successfully done. The DH table of the rover has been determined. As a result, the Homogeneous Transformation Matrices have been found for the arm. The forward and inverse kinematics have been calculated and verified via simulation successfully. The planned task of drilling a hole in the surface of the simulated world has been done successfully, validating the final functionality of the rover.

The learning outcomes that are listed in this report have been covered. We were able to meet our goals. However, we had to fall back to two points of our fallback goals, that is simulating the rover in a simpler world and designing an arm with a lower Degree of Freedom, that is 3. The ambitious goals were not implemented. They are a part of the future work that is planned to be undertaken. We gained a solid understanding of the modeling and control of a rover that can be used for Mars exploration. This project has given us skills that can be used for similar real-time situations and a good opportunity to further our experience with ROS and Gazebo.

REFERENCES

[1] The Planetary Org. *Every Mars Mission*. https://www.planetary.org/space-missions/every-mars-mission. [Online]. 2020.

[2] NASA. *The MSL home*. https://mars.nasa.gov/msl/home/. [Online]. 2020.

[3] fath25. *AachenRover*. https://github.com/fath25/AachenRover. [Online]. 2021.

[4] marsiitr. *Mars Rover*. https://github.com/marsiitr/Mars-Rover. [Online]. 2021.

[5] Youssef Bassil. "Service-Oriented Architecture for Space Exploration Robotic Rover Systems". In: (Apr. 2012).

[6] The Construct. *Mars Rover*. https://bitbucket.org/theconstructcore/curiosity_mars_rover/src/master/. [Online]. 2020.

[7] NASA. *Mars Exploration Program*. https://mars.nasa.gov/resources/25042/mars-perseverance-rover-3d-model/. [Online]. 2020.

Appendix A

MATLAB CODE

Matlab Code for Forward Kinematics validation of Robotic arm

```
clear all
clf
d1=0.222; d3=0.1148; a2 =0.6553;
L(1)= Link([0, d1,  0,  -pi/2, 0]);
L(2)= Link([0,  0,  a2, pi/2, 0]);
L(3)= Link([0, d3, 0, 0, 1],'standard');
L(4)= Link([0,  0, 0,  0, 0]);
L(3).qlim = [0 1];
robot = SerialLink (L);
robot.name = 'Mars Rover';
robot.teach
```