INTRODUCTION TO ROBOT MODELLING

UNIVERSITY OF MARYLAND

DEPARTMENT OF MAGE

Athlete Rover Model

Project 2 Report

Name:

Harshal Shirsath

(*UID: 119247419*)

Tarun Sai Reddy Trilokesh

(*UID:* 118450766)

Professor:

Dr. Reza Monfaredi

December 10, 2022

Contents

List of Figures

List of Tables

1 Introduction

The All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE) is a new mobility platform developed for potential lunar operations. This six-legged rover is designed to perform general tool and payload manipulation, as well as roll quickly over benign terrain, walk over rough and steep terrain, and traverse both types of terrain by rolling and walking respectively. Detailed model and simulation of this platform is presented in this report. The simulation is done using ROS.

Figure 1: Athlete Lunar Rover

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, such as hardware abstraction, low-level device control, implementation of frequently used functionality, message passing between processes, and package management. ROS is not a real-time framework, but real-time code can be integrated with ROS. Additionally, ROS has numerous tools that augment its core functionality. These tools permit developers to visualize, record data, and create scripts, among other capabilities.

rviz - rviz is a three-dimensional visualizer for robots, environments, and sensor data. It is highly configurable, and plugins can be added to expand its functionality.

Gazebo - Gazebo is an open-source 3D robotics simulator. Gazebo was a part of the Player Project between 2004 and 2011. Gazebo incorporated the ODE physics engine, OpenGL rendering, and sensor simulation and actuator control support code.

In this report for ENPM662 Project 2, we detail our process of designing, modeling, and simulating the ”Athlete Lunar Rover” to carry out a simulated task with real-world applications. The report is divided into twenty sections. In Section 2: Application, we will briefly discuss the practical uses for this type of robot and the factors that led us to make this particular choice. In Section 3, ”Robot Type,” we’ll explore the generalized geometric outline of the robot’s structure. The degrees of freedom (DoFs) of the robot will be described in Section 4: DoFs and dimensions based on the movement of the links and the dimensions of each assembly part used to construct the robot. In Section 5: CAD Models, we’ll go over how we modeled the Athlete Lunar Rover in 3D modeling software, complete with screenshots of the various stages and images of the individual parts and the finished product. The Denavit-Hartenburg parameters of the robot model are described in Section 6: DH Parameters, which also features a diagram of the robot in its home position and the derived DH table. Forward Kinematics Section 7 will show how we used the DH Table to derive the robot’s forward kinematics, including the transformation matrices we obtained in a symbolic format for each link with respect to the base frame. In

Section 8, ”Forward Kinematics Validation,” we will detail the steps we took to verify that the forward kinematics we computed in Section 7 for a given set of input joint angularities actually matched the actual positions of the end effectors in the gazebo environment. In Section 9: Inverse Kinematics, we will demonstrate the process by which the inverse kinematics for the Athlete Rover were derived to determine the necessary joint angles to move the end effector to a specified location. Section 10: Inverse Kinematics Validation explains how we checked the derived inverse kinematics for a known end effector trajectory towards a simulated payload. Taking into account the limitations of each link and the overall design of the robot, Section 11: Workspace Study will provide a high-level analysis of the area within which our robot can effectively articulate to complete its tasks. In Section 12: Assumptions, we will detail the various assumptions that informed our approach to the robot's design and operation. In Section 13: Control Method, we'll detail the controllers we used and why we decided to use them for each of the robot's movable parts. The Athlete Rover model walking and running in the gazebo simulation environment will be visualized in Section 14: Gazebo and rviz Visualization through the use of screenshots and external media links. In Section 15: Challenges Encountered, we'll detail the most significant difficulties our team ran into while designing and simulating the robot, as well as the solutions we developed to address them. The most important things we discovered throughout the course of the project will be summarized in Section 16: Lessons learned. The extent to which our team achieved the objectives we set forth at the outset of the project will be briefly discussed in Section 17: Conclusion. In Section 18 future scope for the project is detailed. Validation code for forward kinematics can be found in Section 19's appendix. All of the sources that were consulted during the course of this project's completion are listed in Section 19.

2 Application

One of the most efficient rover and a prototype for Perseverance & Curosity (Mars rovers).The Athlete rover is a multi purpose system capable of docking or mating with special purpose devices including refueling stations, excavation equipments and different special end effector. It is a 6-Dof manipulator leg for Wide Range Of Terrains.

The six 6-DOF legs allow more capabilities than other robotic systems such as Sojourner or the Mars Exploration Rovers. These mean that the slopes it could climb would be up to 35° on solid surfaces and 25° on soft surfaces, such as the soft deposits of dust found on the Moon. Plans are to develop the system's capability of travel over rougher terrain and to increase the speed of ATHLETE to 10 kilometres (6.2 mi) per hour, 100 times faster than the Spirit and Opportunity rovers.

This adaptable robotic platform will serve as a mobile base for pressurized lunar habitats, enabling long-range surface exploration as well as the transport of crew members. In-situ construction of lunar assets will also be made possible, giving astronauts the ability to assemble, maintain, and service a wide variety of cargo.

3 Robot Type

Athlete is a Lunar Rover. It consists of 6 legs which act as manipulators and each of these manipulators are of 6 Degrees of Freedom. Since, considering the overall environment and the surface content on the moon these rovers come with a much stable system for it to be more maneuvering with a much slower speed for a better

span and achieving all terrain ranges. The multipurpose

Figure 2: Athlete Lunar Rover Multi-Agent System

ATHLETE vehicle is composed of six limbs with six joints each. Each of these limbs is identical and has a yaw, pitch, pitch, roll, pitch, roll kinematic structure as illustrated in Figure 2. The end effector of each limb is a powered wheel which can be used on gentle terrain to drive at speeds up to $10\mathrm{kph}$. On rough and steep terrain the wheel can be locked allowing the vehicle to walk.

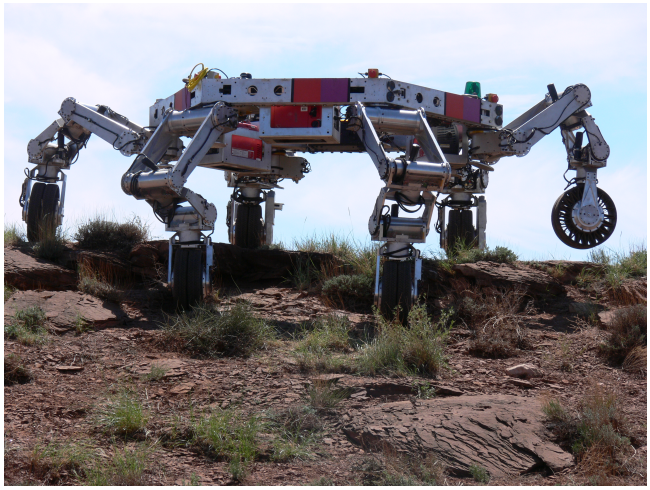


Figure 3: Athlete Lunar Rover Traversing in Rough Terrain

4 DOFs and Dimensions

The $850\ \mathrm{kg}$ ATHLETE vehicle (model weight $284\ \mathrm{kg}$) is designed with an open hexagonal frame, as shown in Figure 4, that measures $2.75\ \mathrm{m}$ (model weight $2.55\ \mathrm{m}$) tip to tip with a six degree of freedom limb located at each vertex of the hexagon which gives the rover 36 degrees of freedom overall.

This axisymmetric robot was designed as a prototype vehicle to test the wheel-on-limb mobility concept as well as provide a platform for software development. The research nature of the project demands spaceflight design principles without requiring the use of spaceflight hardware. This allowed for the rapid design, build, and test cycle of the vehicle.

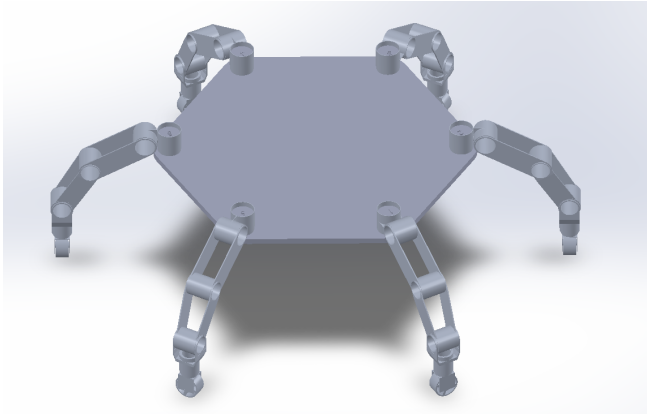


Figure 4: Athlete Lunar Rover CAD

5 CAD Model

The rover is designed using solidworks software. Solidworks is a solid modeling computer-aided design (CAD) and computer-aided engineering (CAE) application published by Dassault Systèmes. Using solidworks all the joints and frame is modelled with accuracy. Solidworks offers URDF export functionality which is very essential to model and simulate the rover in ROS. To do the following the the frame is considered to be the base link and the limbs are considered to be the child links. The joints are mated to each other and the coordinates are mentioned (coordinates of the joints are important) with the coordinates the joint rotation axis also is mentioned. The mated joints are constrained with the upper and lower angular limits. The model is then exported using the solidworks addon sw2urdf. During the export previously mentioned coordinates and axes are configured respectively. The links and axes of all the joints are named which will be used during ROS simulation.

6 DH Parameters

Kinematics analysis is the basis of gait planning and attitude control of a hexapod robot. Kinematics studies all geometric and time parameters related to motion. To complete the kinematic analysis of the robot, the first problem is to describe the robot parametrically. Based on standard Denavit-Hartenberg (D-H) notation, the Athlete leg links from the figure 5 can be described as in Table 1.

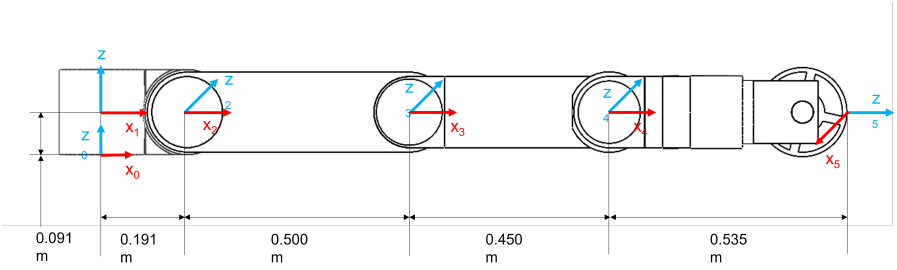


Figure 5: Athlete Leg

In Table 1, a is the link length, α is link twist, d is the offset, θ is the joint angle.

J	a_i	α_1	d_i	θ_i
1	0	0	0.091	θ_1
2	0.191	$-\pi/2$	0	θ_2
3	0.500	0	0	θ_3
4	0.450	0	0	θ_4
5	0.535	0	0	θ_5

Table 1: DH parameters

7 Forward Kinematics

In robot kinematics, forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters.

The kinematics equations of the robot are used in robotics, computer games, and animation.The kinematics equations for the series chain of a robot are obtained using a rigid transformation [Z-axis] to characterize the relative movement allowed at each joint and separate rigid transformation T_{i-1}^{i} to define the dimensions of each link. The result is a sequence of rigid transformations alternating joint and link transformations from the base of the chain to its end link, which is equated to the specified position for the end link:

The specified DH parameters θ_i, d_i, a_i and α_i are the rotation around z, translation along z, translation along x and rotation around x, respectively. The homogeneous transformation matrix describes the relationship of a point in the different coordinate systems. The homogeneous transformation matrix from the foot end coordinate system to the reference coordinate system can be obtained by:

$$
\begin{aligned}
H_{i+1}^{i} &= \mathrm{Rot}_{z,\theta_i} \cdot \mathrm{Trans}_{z,d_i} \cdot \mathrm{Trans}_{x,a_i} \cdot \mathrm{Rot}_{x,\alpha_i} \\
&= \begin{bmatrix}
c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\
s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\
0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{aligned}
$$

where c_x and s_x denote $\cos(x)$ and $\sin(x)$ respectively.

In the notation H_{i+1}^{i}, the superscript denotes the reference frame $(o_i x_i y_i z_i)$ and the subscript indicates the transformed frame $(o_{i+1} x_{i+1} y_{i+1} z_{i+1})$. Using Fig. 4 as a 36 DOF hexapod example, for a leg of the robot the transform from the leg frame $(o_1 x_1 y_1 z_1)$ to the end effector $(o_e x_e y_e z_e)$ (wheel) is given by:

$$
H_e^1 = H_2^1(q_1) \cdot H_3^2(q_2) \cdot H_4^3(q_3) \cdot H_5^4(q_4) \cdot H_e^5(q_5)
$$

where q_1, q_2, q_3, q_4, and q_5 are the joint angles for the coxa $_{yaw}$, coxa, femur, tibia, pitch and roll joints respectively.

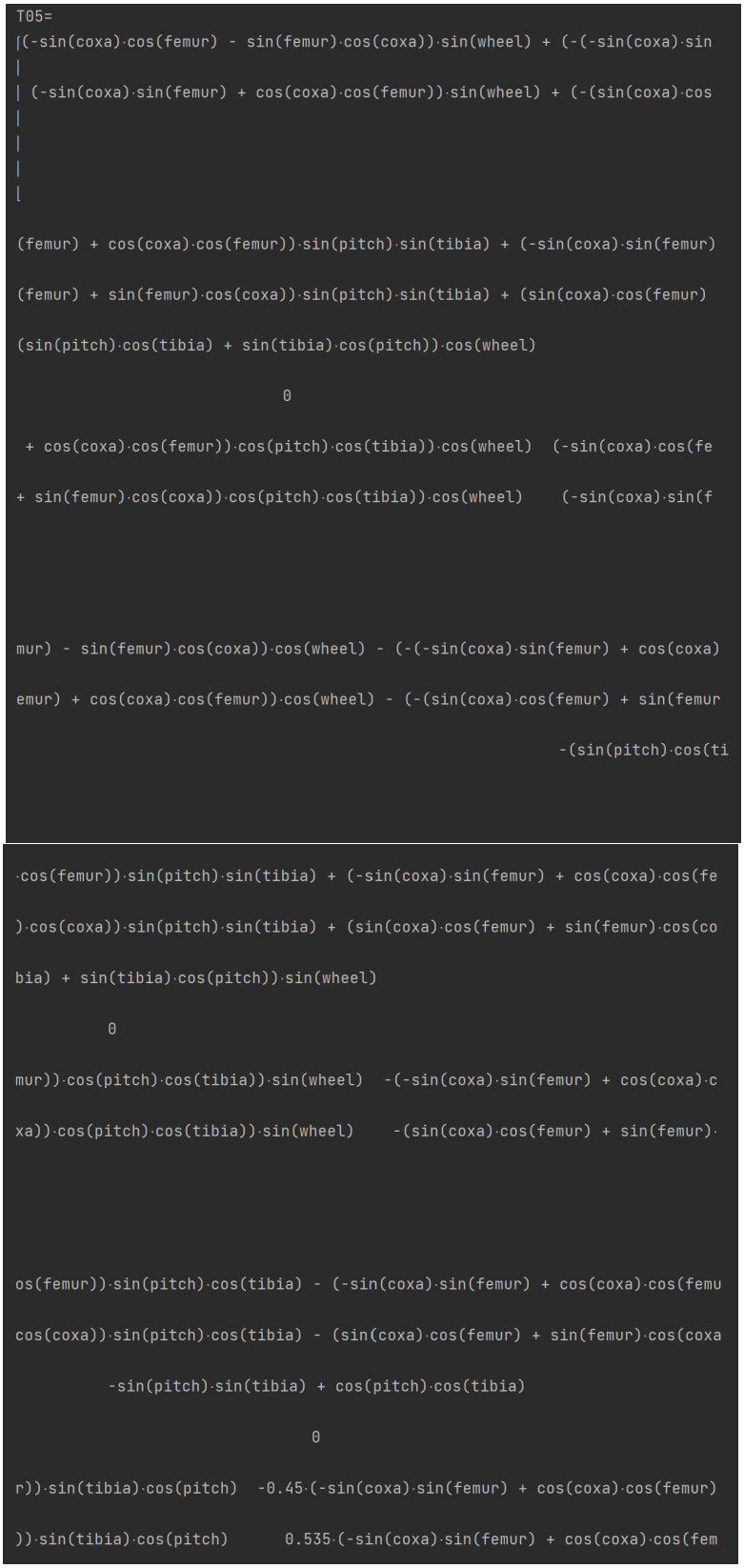


Figure 6: Transformation Matrix

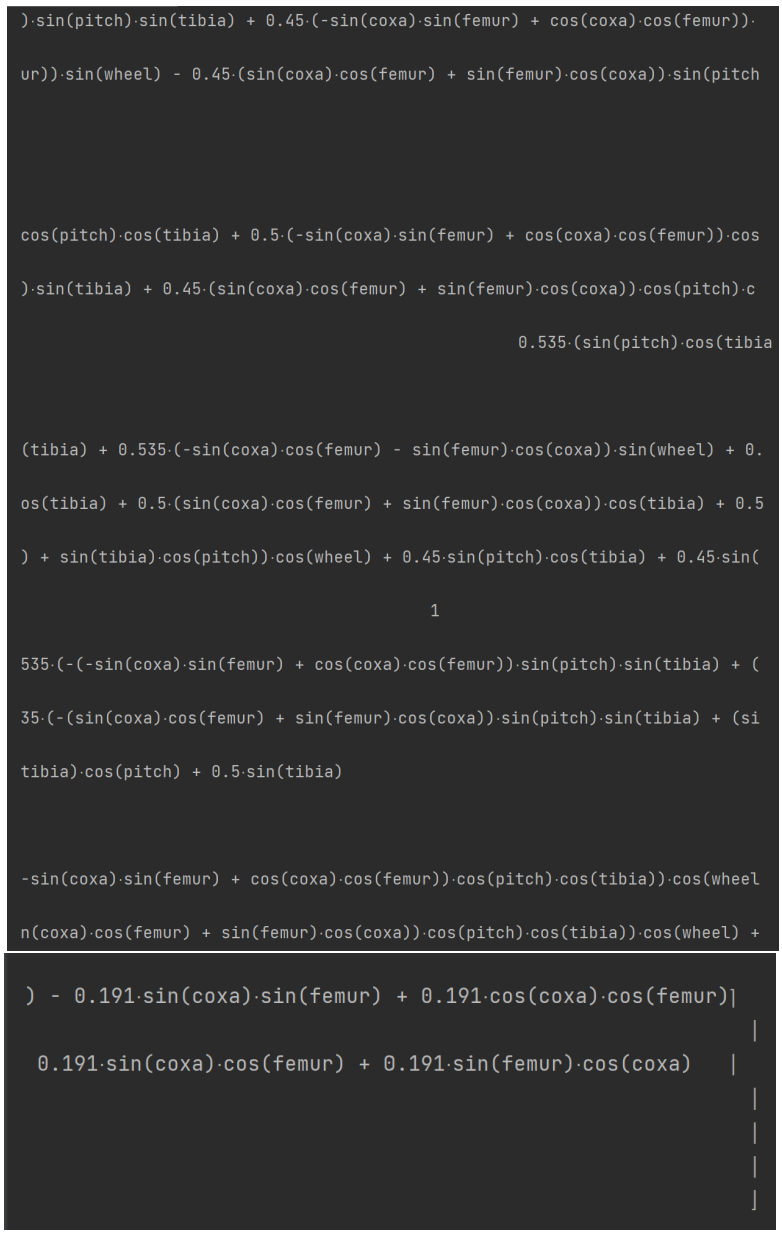


Figure 7: Forward Kinematic Transformation

8 Forward Kinematics Validation

As mentioned previously in the Forward Kinematics the joint angles are fed in the robot and we obtained the position of the end-effector. To validate this we derived the transformation matrices of each joint with respect to the base frame with the help of the D-H table.

The joint angles are provided to all the legs and the robot follows the as expected and maintains all the wheel perpendicular to the ground. This can be seen the simulation. To see the video click the Link.

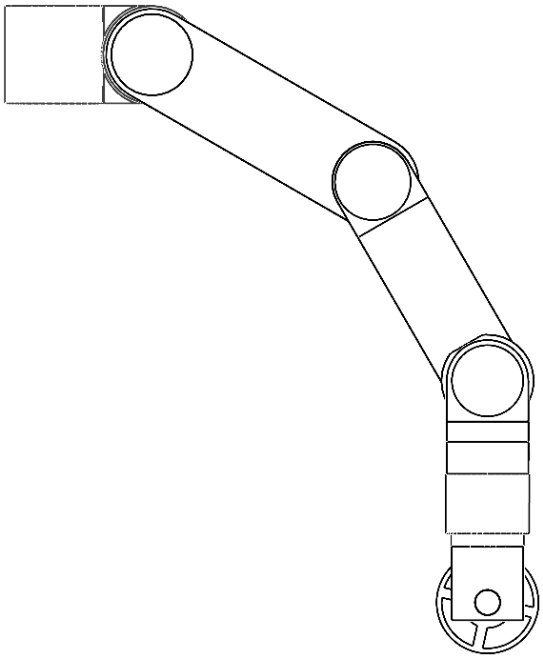


Figure 8: Athlete Leg Extended

Figure 8 gives the reference for the leg when extended. The angles are taken from solidworks by keeping the wheel parallel to the ground. These angles are then fed to the robot during simulation and the robot performs as needed. We also calculated the forward kinematics of the leg when extend and the coordinates the verified. Forward kinematics of the end effector (wheel in this case) is calculated using the code attached in the appendix 20.1 (also provided python file with the report for validation). The output from the code is obtained as seen in figure.

```
End Effector Tansformation Matrix=
⎡ 0.43301270189222         -0.75               -0.5          0.966928363486664⎤
⎢                                                                             ⎥
⎢       0.25         -0.433012701892219  0.866025403784439  0.55825635094611 ⎥
⎢                                                                             ⎥
⎢-0.866025403784439         -0.5                 0          -1.01203502272767⎥
⎢                                                                             ⎥
⎣       0                    0                   0                 1.0       ⎦
```

Figure 9: Extended leg Transformation matrix

9 Inverse Kinematics

In robotics, inverse kinematics is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain, such as a robot manipulator, in a given position and orientation relative to the start of the chain. Given joint parameters, the position and orientation of the chain's end, e.g. the hand of the character or robot, can typically be calculated directly using multiple applications of trigonometric formulas, a process known as forward kinematics. However, the reverse operation is, in general, much more challenging.

To calculate the inverse kinematics for the rover individual legs were considered. Using the DH parameters in the section 6 the robot leg retracts to predefined position and rotates along the coxa, these steps are repeated in a sequence and keeps the frame as stable as possible. The joints (coxa, femur, tibia, pitch and roll) on the leg when extended maintain $0, 30, 30, 30, 0$ degrees respectively and when retracted maintain $0, -60, 90, 30, 0$ degrees.

10 Inverse Kinematics Validation

To validate the inverse kinematics we have designed gait system for the rover. The gait system used is single leg gait system. This keep the rover very stable and moves the rover at desired speed. The whole system works in a sequence and the robot obeys the theoretically calculated trajectory, which can be seen in the video Link. This gait system was designed by trail and error method and found that single leg gait system is most suitable for this frame and application. The robot moves at speed of $30\ \mathrm{m/s}$ approximately to reacts the goal as defined.

11 Workspace Study

The set of places that the rover's end-effector can reach is its workspace (wheels). In the aforementioned situation, the vicinity of the rover within which it can extend its legs and do the necessary activities might be said.

The dimensions of the robot, the types of joints present between the links, and the restrictions on movement placed upon each joint are all taken into account while calculating workspace.

As stated earlier, the athlete has six legs, each with five revolute joints and one continuous joint. The first revolute joint is the coxa, and its range is from -120 to 120 degrees. All other revolute joints (femur, tibia, pitch, and roll) have a range of -90 to 90 degrees. The wheel joint is controlled using velocity, so its rotation is continuous around the wheel axis.

12 Assumptions

The NASA Athlete Lunar Rover has a 6 legged features robot and in a way more tedious as compared to normal driven robots. Therefore, while computing this robot there were a lot of many assumptions which were considered. These are presented in the following points:

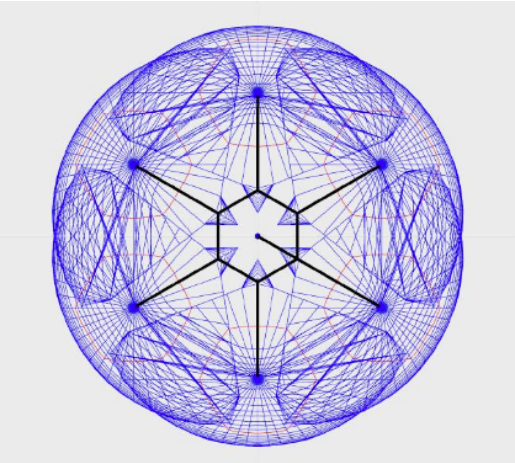


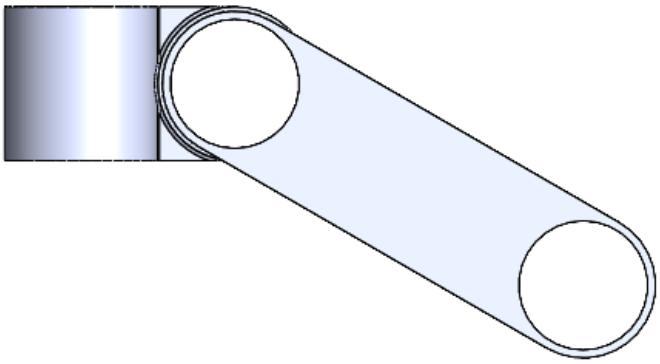
Figure 10: Workspace of Athlete Rover

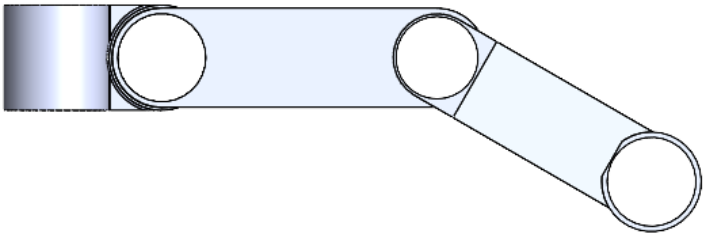
- The joints are well equipped with the infinite torques required. This means the other component weights were not considered in this simulation. Basically, it just consisted of the external structure.
- The actual friction components on the joints came out through trial and error. While, in the actual scenario the situation would be totally different.
- When the robot starts crawling the wheel rotation were locked, in order to act as a rigid link and no rotation on any of the wheels with a considerate friction between the wheels and the ground.

13 Control Method

According to the links present in the robot, there were 36 joints as mentioned in the introduction. Firstly, there are pitching links, yaw links, and & roll links in every Athlete Lunar Rover leg. In this particular robot, we have just tried to achieve all the tasks by using just velocity and effort controllers on the respective joints.

- There are 3 pitching links in one leg. These joints are at the effort controllers and provide some torque to the link.





- While there is just 1 yaw joint linking the leg and the body. This joint is defined as an effort controller.
- Adding further, there are two roll joints one for the wheel and other for the steering of the wheels. In these, we see that the wheel has the velocity controller whereas the steering-wheel joint has an effort controller. So, overall, if we are considering all the joints and their respective controllers, here they are:

Joint Number	Joint Name	Joint Controller
1	Coxa	Effort Controller
2	Femur	Effort Controller
3	Tibia	Effort Controller
4	Pitch	Effort Controller
5	Roll	Effort Controller
6	Wheel	Velocity Controller

Table 2: Leg Joint Controllers

Serial Number	Type of Controller	Total Number
1	Velocity Controller	6
2	Effort Controller	30

Table 3: Number of Controllers

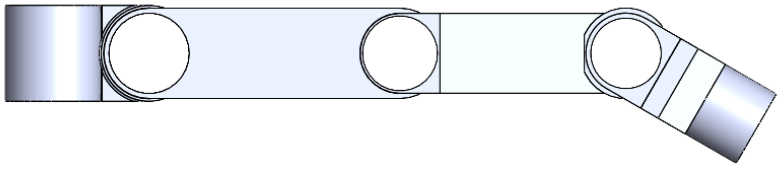


Figure 11: Pitch

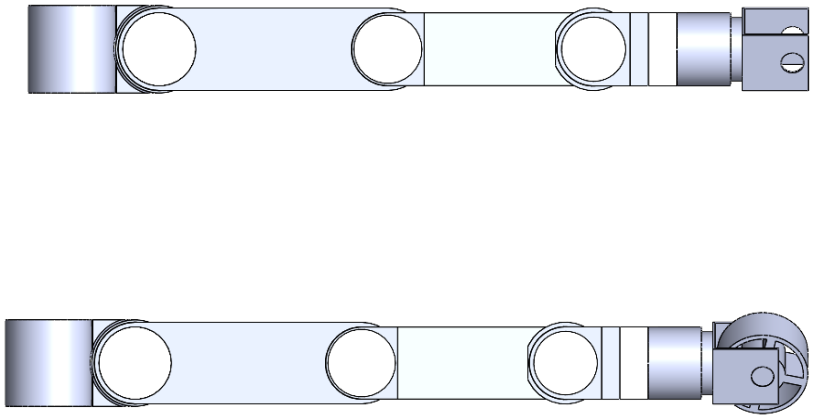


Figure 12: Roll

14 Gazebo and Rviz Visualization

The rover is modelled and simulated in Gazebo and rviz. In the rviz video, we can see that the robot joints are moving as planned, and to simulate this joint state publisher gui is used. By performing this, it is validated that all the joint coordinates and axes are defined. This is also done to verify that the home position of the joints is as desired. Figure 13 shows the model simulated in the rviz and controlled using the joint state publisher gui. This can be seen in the video. To see the video, click the Link.

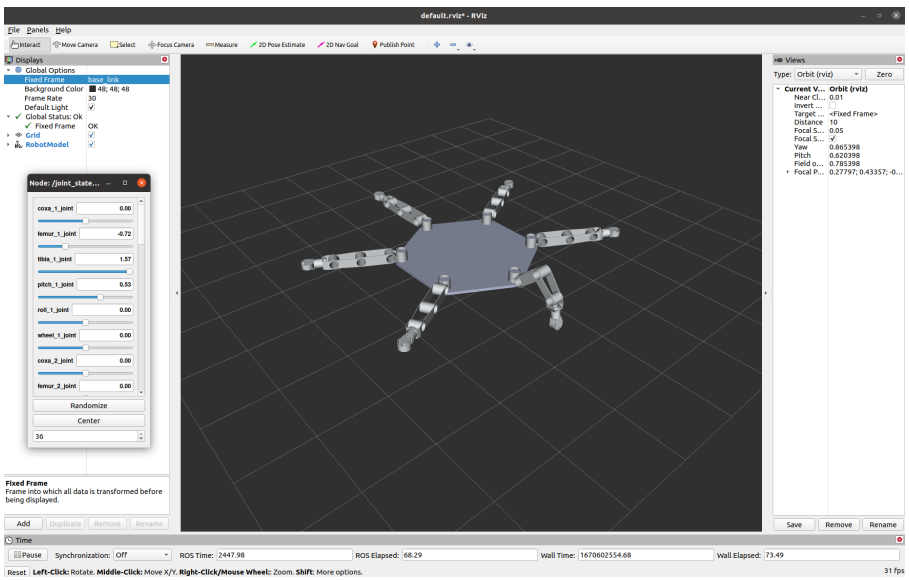


Figure 13: Athlete Rviz Simulation

The robot is also simulated in gazebo; as seen in figure 14 the rover is standing, and these functions are controlled by running athlete_controller.py in the scripts directory of the package. To see the video, click the Link. The rover can do a variety of things, including storing for transportation purposes, homing joint positions, standing up, turning right, turning left, moving forward, moving backward, and walking.

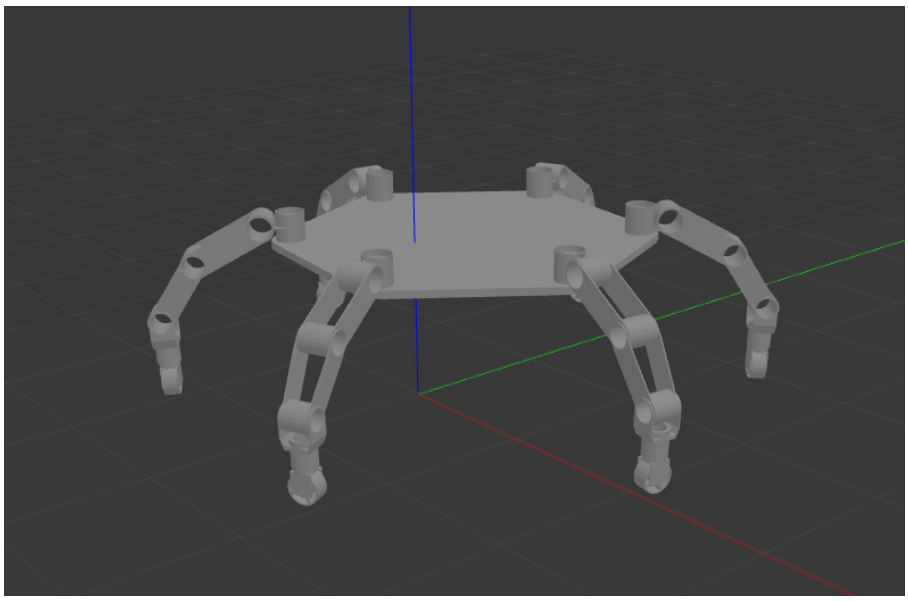


Figure 14: Athlete Gazebo Simulation

15 Problems Faced

There were many challenges that we faced during this project.

- One of the major challenges we faced was spawning the rover in Gazebo using the urdf file generated from SolidWorks. The joints did not maintain the right coordinates and axes, which had to be corrected manually.
- We also had trouble assigning controllers to the joints, and this was done with the trial and error method.
- Considering the weight of the platform, the joints faced difficulty in lifting the rover, which required PID tuning. Each joint was PID tuned to get the rover off the ground.
- Designing the gait system for the rover was challenging. The rover faced issues keeping balance and moving in a straight line.

16 Lessons Learned

Projects with integration come up with a lot of debugging and solving minute errors. Likewise, we came across the same domain of mistakes which were worth learning during this project phase.

- URDF manipulation has to be perfect otherwise the spawning the robot in the Gazebo world can be very infuriating. Therefore, it should be made use the set position of the links and their mates are provided with proper constraints.
- Implementing the controller and their respective axes is also one of the vital part which takes a lot of time in getting debugged. In our case it became even more tedious since the number of joints were around 36 in all.
- Setting up the force parameters to the effort controller in order to make sure the robot when spawned in the world does not collapses with the base ground frame of the world.

17 Conclusion

The project on NASA Athlete Lunar Rover was one of the most incredible project ever worked upon. It came up with various types of challenges involved even from the basic export of part file into a gazebo world. It really taught us how defined a package has to be in a particular way. Spawning the file in Gazebo world and constraining the joint angles with specifying controllers. The most vital feedback we would like to give about this project is it gave us the overall scenarios regarding simulations, integrating all the elements from all perspectives and to plan out the methodology to approach every problem much more precisely.

18 Future Work

The robot has the multiple functions with respect to certain applications. This robot is well designed to overcome all the range of terrains. However, there are many other way in which the future scope of an Athlete Lunar rover could be used.

- The robot comes with a capability in attaching multiple types of end-effector. For example, ploughing claw, robotic gripper etc.

Figure 15: Types of End Effector

- The robot could be used for the swarm robotics application to carry out larger payload.
- Adding further, implementation of cameras, lidar sensors to scan the environment.
- Docking payload of 250kgs.

19 References

[1] Zhuang H-C, Gao H-B, Deng Z-Q. Gait Planning Research for an Electrically Driven Large-Load-Ratio Six-Legged Robot. Applied Sciences. 2017; 7(3):296. https://doi.org/10.3390/app7030296

[2] B. Tam, F. Talbot, R. Steindl, A. Elfes and N. Kottege, ”OpenSHC: A Versatile Multilegged Robot Controller,” in IEEE Access, vol. 8, pp. 188908-188926, 2020, doi: 10.1109/ACCESS.2020.3031019.

[3] Liu Y, Fan X, Ding L, Wang J, Liu T, Gao H. Fault-Tolerant Tripod Gait Planning and Verification of a Hexapod Robot. Applied Sciences. 2020; 10(8):2959. https://doi.org/10.3390/app10082959

[4] Zhai S, Jin B, Cheng Y. Mechanical Design and Gait Optimization of Hydraulic Hexapod Robot Based on Energy Conservation. Applied Sciences. 2020; 10(11):3884. https://doi.org/10.3390/app10113884

20 Appendix

20.1 Forward Kinematics

```
from sympy import symbols, Matrix, eye, sin, cos, pi, pprint, diff
import math
import sympy as sym
import numpy as np
import matplotlib.pyplot as plt
from functools import partial
from mpl_toolkits.mplot3d import axes3d, Axes3D



Q1, Q2, Q3, Q4, Q5= symbols('coxa femur tibia pitch wheel')
Joint_Angles = [Q1, Q2, Q3, Q4, Q5]

round_3 = partial(round, ndigits=3)
t = 0

DH_Parameter = [
    {'a': 0, 'd': 0.091, 'alpha': 0      },
    {'a': 0.191, 'd': 0, 'alpha': -pi / 2},
    {'a': 0.500, 'd': 0, 'alpha': 0      },
    {'a': 0.450, 'd': 0, 'alpha': 0      },
    {'a': 0.535, 'd': 0, 'alpha': 0      },
]

FKine = eye(4)
Jacobian_M = eye(6)
T = []
Z_M = []
O_M = []

for i, (P, Q) in enumerate(zip(DH_Parameter, Joint_Angles)):
    d = P['d']
    a = P['a']
    alpha = P['alpha']

    Transform_M = Matrix([[cos(Q), -sin(Q) * cos(alpha), sin(Q) * sin(alpha), a * cos(Q)], \
                          [sin(Q), cos(Q) * cos(alpha), -cos(Q) * sin(alpha), a * sin(Q)], \
                          [0, sin(alpha), cos(alpha), d], \
                          [0, 0, 0, 1]])

    T.append(Transform_M)
    FKine = Transform_M @ FKine

    Z_M.append(FKine[0:3, 3])
    O_M.append(FKine[0:3, 2])

T01 = T[0]
```

```
48 T02 = T[0] * T[1]
49 T03 = T[0] * T[1] * T[2]
50 T04 = T[0] * T[1] * T[2] * T[3]
51 T05 = T[0] * T[1] * T[2] * T[3] * T[4]
52
53 print("End Effector (wheel) Tansformation Matrix=")
54
55 Transform_M_end_eff = T05.subs({Q1: math.radians(0), Q2: math.radians(30), Q3: math.radians(30), Q4:
       math.radians(30), Q5: math.radians(0)}).evalf()
56 pprint(Transform_M_end_eff)
```