ENPM662 Project2 Report

Quadruped Robot with Parallel Manipulators

Submitted on December 9th, 2020

Authors:
Jayesh Jayashankar (UID: 117450101)
Charu Sharma (117555448)

Table of Contents

Abstract

There are many innovative ways in which quadruped robots are enhanced to perform different operations such as delivery, pick & place, navigating through difficult terrain etc. In this project focus on enhancing a 4-legged robot with additional features of parallel manipulators along with a platform to mount objects. We develop the CAD model of the robot through SolidWorks and export the URDF as a ROS package. The forward kinematics of the robot is simulated in RVIZ using the joint state publisher GUI and validated with the end effector coordinates obtained through a python script. The inverse kinematics is validated by running a python script which computes the respective joint angles for the end effector positions previously obtained. For the gait analysis we focused on a crawl gait and a trot gait which we were able to successfully simulate in Gazebo. We also present the challenges we came across during the simulations and the scope for future work.

1. Introduction

Legged robots are commonplace over the last few years, and a lot of companies and educational institutions are looking to develop these robots in versatile ways. The most common types of legged robots are bipedal & quadrupedal. These are particularly preferred because the bipedal model emulates the human form, and the quadruped model emulates the animal form. Some of the applications for these robots include delivery operations & surveillance. These robots are well suited to perform these complex tasks since these applications require maneuvering across uneven terrain, integration with different sensors and cameras, and the ability to achieve high mobility & dexterity.

There are different methods with which delivery of goods are accomplished. Drone deliveries are services which companies such as Amazona and Walmart are looking to offer soon. Amazon through Prime Air and Walmart through its partnership with zipline are the services which would significantly reduce the delivery time over long distances. Legged or wheeled robots fulfil the purpose of short distance deliveries. ANYbotics [1] uses quadruped robots which complete last mile deliveries by travelling in an autonomous vehicle and walk up to the destination carrying the packages on a platform. Bipedal robots such as Boston Dynamics's Atlas [4], a humanoid uses the arms to carry objects. Wheeled robots such as Kiwibot and Marble are used for food delivery and post mail delivery, respectively.

There has also been work carried out to use the legs as arms for performing pick and place operations. RoMeLa's ALPHRED2 [3] quadruped robot uses two legs to pick up objects and mount them on a platform on the robot. The robot would balance itself on the remaining legs whenever a task is being performed by the other legs. ANYbotics's robot can balance itself on 3 legs by ringing the doorbell with its foot carrying parcels of up to 10 Kg. Boston dynamic's spot [2] can be customized to include a manipulator which can perform tasks such as opening doors, picking and placing objects.

From the work carried out in the different models in [1], [2], [3], [4] integrating a quadruped robot with different functionalities increase its capabilities. Through this project we aim to collaborate different ideas presented above and add an additional layer of functionality by using two arms and an object mounting platform on a 4-legged robot. Thus, the quadruped robot will not only be capable of walking efficiently across different paths but also be capable of carrying heavy load and use the arms to perform various operations such as pick and place, clearing obstacles in the path, ring doorbells etc.

The report has been structured as follows:
1. Motivation behind the robot model
2. Description of the robot model
3. Robot appropriateness for the task
4. Scope of the study
5. Assumptions
6. Forward Kinematics
7. Inverse Kinematics
8. Gait Analyses
9. Software tools used
10. Validation
11. Discussion and Conclusion
12. Scope for future work

2. Motivation

We investigated different models and identified elements or functionalities lacking in these systems and based on that we came up with a model plan which can add that extra dimension to a quadruped robot. For example, ANYBOTICs ANYMAL is a quadruped robot capable of carrying load up to 10 kg on its platform. The robot drops the package in the customer's doorstep by tilting its platform. This can be a drawback if the package consists of fragile items. Another example is Boston Dynamics's spot with a manipulator. This robot can perform manipulator operations with a single arm but falls short when the requirement is to carry heavy objects or large objects due to the lack of a second arm. RoMeLo's alphred2 solves this purpose by using two arms to pick up objects while balancing on its other legs and places the object on its platform or in a destination location. But this robot again falls short when it comes to lifting heavier objects since the system's stability would get reduced when balancing on 2 or 3 legs. Wheeled robots such as KIWIBOT and MARBLE require the customer to pick up the parcel from the robot which requires the customer to be physically present to pick up their parcel.

As part of our study, we want our system to be capable of achieving the following
(i). Walk across different paths such as stairs, uneven terrain, slopes etc.
(ii). Have two manipulators which work together or separately to perform different manipulator operations such as pick and place, carrying heavy load, ringing doorbell etc.
(iii). Consist of a platform to mount heavy objects to be carried to the destination.

With these characteristics the drawback of the ANYMAL robot which drops the package can be avoided by using parallel arms to place objects safely. This model can also overcome the shortcomings of ALPHRED2 as the system will be more stable and can carry more weight since the legs are not going to be used to perform any other operation. The other advantage is that the customer does not have to be physically present to receive the delivery which is in the case of KIWIBOT and MARBLE.

Therefore, we present the quadruped robot with parallel manipulators. The arms would be used to pick and place the objects on this platform or onto a destination location. This ensures that the quadruped robot is not only capable of navigating across uneven terrain but also has the ability to perform multiple functions. This added functionality can prove to be advantageous not only in delivery operations but also in industrial and military applications.

3. Robot Description

The design of the robot can be divided into 3 parts
(i) **Quadruped**: the 4 legs would have 2 DOF one at the hip joint and another in the knee joint connected to the robot body.
(ii) **Manipulators**: two 3 link arms where each arm would have 3 DOF with an appropriate end effector to grip the object.
(iii) **Platform**: Fixed rectangular platform on top of the robot to mount objects

Figure 1 shows the Solid works model of the model for the robot consisting of the parts specified above.

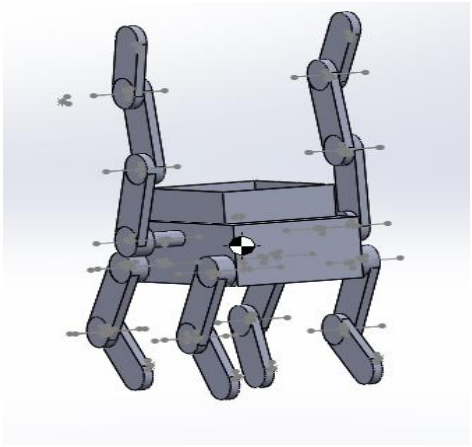

Figure 1: SolidWorks Model of the robot

Design of the quadruped part of the robot

For this part of the robot, we designed a base or the body which would contain the required controllers, battery, and other electronics. The legs of the robot is connected to the left and right side of the body through the hip joint. There would be two joints one in the hip and the other in the knee and all the joints are revolute in nature. These joints would be driven by torques generated by DC motors. There will be 2 links one from the hip joint to the knee joint and the other from knee joint to the ground. The parameters we used are as follows in Table 1 & 2:

Table1. Leg parameters

Parameter representations	Parameter description	Values
l1	length of link from hip joint to knee joint	5.0 (inches)
l2	length of link from knee joint to the ground	3.5 (inches)
$\theta 1$	Joint parameter across the hip joint	(-180,180)
$\theta 2$	Joint parameter across knee joint	(-180,180)
b1	Length of base or body of the robot	10.0 (inches)
b2	Breadth of base of the robot	7.0

Table2. Arm Parameters

Parameter representations	Parameter description	Value
a1	Length of link connecting shoulder joint to elbow joint	5
a2	Length of link connecting elbow joint to wrist joint	5
a3	Length of link from wrist to end effector	3.5
$\theta 1$	Joint parameter across the shoulder joint	(-180,180)
$\theta 2$	Joint parameter across knee joint	(-180,180)
$\theta 3$	Joint parameter across knee joint	(-180,180)

Design of Platform to mount objects

This platform is setup on top of the body of the robot. This area would be used to place the objects and walk along with it to the destination. We intend on going with a design similar to the ANYMAL's robot. We have added a hollow cuboid which sits atop the body of the robot. The parameters are as described in Table 3.

Table 3. Platform Parameters

Parameter representations	Parameter description	Value
b1	Length of the platform	7.0
b2	Width of the platform	6.0

4. Robot Appropriateness for the task

The scope for delivery robots keeps expanding and thus we need to see what sort of features can be added to the robot to make sure it is as good or better than its human counterpart. A package delivery task would involve the delivery person carrying the parcel, walking to a customer's home through different terrain such as stairs, slopes, or other obstacles in the way and handing over the parcel or placing it in a specific area. In the proposed model, the 4-legged robot ensures that it is capable of navigating to the destination. The platform on the robot is the place where the package resides thus allowing the robot to carry single/multiple packages. The arms play the crucial role of mounting the package on the platform or handing it to the customer or placing it on the doorstep in the absence of the customer.
Since other delivery robots require the use of one of the legs to perform another activity, the chances of instability in the system are quite high. Therefore, this robot is appropriate to handle all the tasks required in a delivery process.

5. Scope of Study

The tasks which we intend to complete are as follows:

Task1: Creates a 3d SolidWorks assembly of the different parts of the robot. The different parts of the robot include the legs, the body of the robot, the manipulators, and the platform.
Task2: Exported the model as a URDF by defining the axes for the different joints as depicted in the above diagram.
Task3: Created a file to launch the robot in RVIZ by establishing relationships as in the URDF file to perform forward kinematics
Task4: Obtained the end effector values in the RVIZ world frame for all the legs by publishing different angles using the joint state publisher GUI.
Task5: Developed a python script to validate the forward kinematics by creating a DH table and computing the transformation matrices as in the RVIZ world frame.
Task6: Validated the forward kinematics of the end effector positions obtained in RVIZ with the values obtained through the python script for different joint angles and observed the differences between the two.
Task7: Developed a script to compute the inverse kinematics. Created a publisher script which publishes the end effector positions and a subscriber script to compute the joint angles.
Task8: Validated the angles obtained through the scripts with the angles for the different end effector positions in RVIZ.
Task9: Developed scripts to implement gaits for the robot in gazebo. Walk and trot gaits were analyzed and simulated.

6. Assumptions

The various assumptions we considered are as follows:
1. The links of the robot are assumed to be rigid bodies.
2. The robot already has the information of the path and the trajectories to follow.
3. The torques applied at the actuators in the joints of the robot are large enough to provide motion.
4. There will not be any interference between the motion of the manipulators (if we implement am kinematics) and the legs of the robot
5. The dynamics of the robot have already been implemented.
6. The robot is capable of detecting obstacles through sensors such as lidar and make adjustments in its motion according it.
7. The motion of the robot would be in a 2-d direction along the X-Y plane.
8. There will not be any friction acting on the joints and there is enough friction between the legs and the ground to prevent it from slipping.

7. Forward Kinematics

To perform the forward kinematics in each leg we need to consider the frames as in the URDF file. There are 4 parts in each leg, the hip joint, hip link, knee joint and the knee link. The hip joint is the part connected to the robot base and the hip link, the hip link connects the hip joint and the knee joint, and the knee joint connects the hip link with the knee link. We have setup another link called a dummy link which is at the foot piece of the robot as the end effector. So, in total there are 5 frames which are oriented in different ways. While setting up the DH table and the transformation matrices we need to consider how each frame is oriented with respect to the other

7.1 Left Front Leg

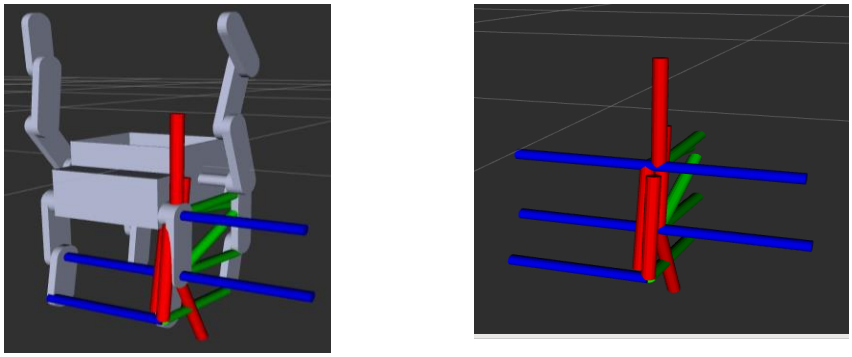

Figure 2: Frame assignment in RVIZ with and without robot model

From the above figures for the left front leg, we can see 5 frames in total. The red blue representing the Z axis, the red representing the X axis and the green representing the Y axis. This is because there are two frames in the hip and knee joint and a dummy frame in the foot (end effector position).

Therefore, based on these frames, the position and orientation of these frames the DH table would be computed.

The origin o_0^0 in this case as in the relative position of the zeroth which is [-0.1016 -0.0889 -0.07874] and the orientation RPY is [-1.5708 1.4089 3.1416]. Therefore, based on this pose the origin of frame 1 should be obtained.

$$o_0^1 = o_0^0 + dz_0^0 + ax_1^0 \longrightarrow 1$$

$$z_0^0 = \begin{pmatrix} \sin(\emptyset)\sin(\varphi) + \cos(\emptyset)\sin(\theta)\cos(\varphi) \\ -\cos(\emptyset)\sin(\varphi) + \sin(\emptyset)\sin(\theta)\cos(\varphi) \\ \cos(\theta)\cos(\varphi) \end{pmatrix} \longrightarrow 2$$

where $\emptyset$, θ, φ refers to the roll, pitch, and yaw angles with respect to the zeroth frame.

$$x_1^0 = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix} \longrightarrow 3$$

where θ in this case refers to the joint angle.

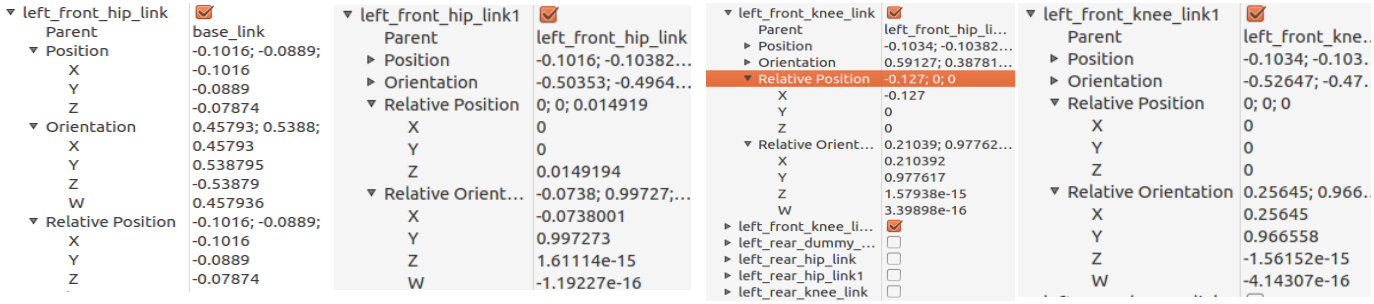

Figure 3: Relative positions and orientations of each frame with respect to another

Based on the relative positions and orientations of the subsequent frames the DH table is obtained.

Table 4: DH table of left front leg

i	*α*	*a*	θ	*d*
1	-171.3	0	π+theta1	0.01491
2	155.72 deg	-0.127	π	0
3	150.25 deg	0	-π + theta2	0
4	0	-0.0895	0	0.0022

The transformation matrices are of the form:

$$A_j^i = \begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow 4$$

The final transformation matrix can be obtained as

$$T_n^0 = \prod_{i=0}^{n-1} A_{i+1}^i \longrightarrow 5$$

The position vector representing the end effector coordinates can be obtained as:

$$p^0 = T_n^0 p^n \longrightarrow 6$$

where $p^0 = [x\ y\ z\ 1]^T$ and $p^n = [0\ 0\ 0\ 1]^T$

7.2 Left Rear Leg

The below figures depict the frames assigned in the left rear leg. The DH table similarly needs to be created based on the position and orientation of each frame with respect to the other.

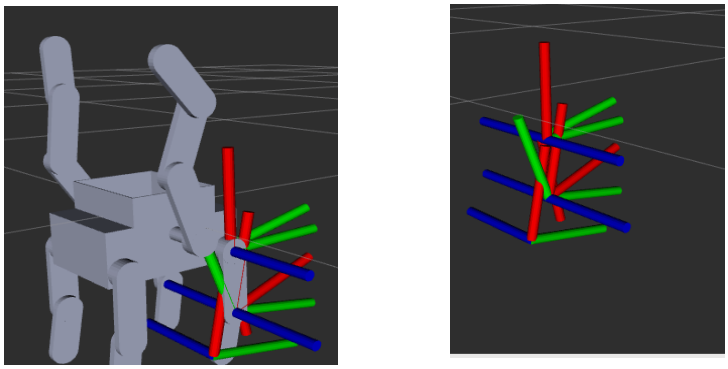

Figure 4: Frame assignment in the left rear leg

There based on these frames and the position and orientation of these frames the DH table would be computed.
The origin o_0^0 in this case as in the relative position of the first frame is [0.1016 -0.0889 -0.07874] and the orientation RPY is [1.5708 1.3266 0].
Therefore, based on this pose the origin of frame 1 should be obtained. Similarly based on the relative positions and orientations of the subsequent frames the DH table is obtained.

Table 5: DH table of left rear leg

i	*α*	*a*	θ	*d*
1	-167.4deg	0	- π + θ1	0.0155
2	70.41 deg	-0.127	Pi	0
3	58.49 deg	0	π + θ2	0
4	0	-0.0895	0	0.0028

The transformation matrices are obtained by substituting these values in equation 4 to obtain the end effector coordinates.

7.3 Right Front Leg

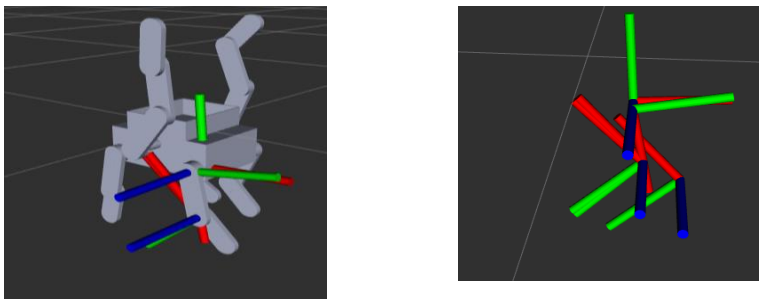

Figure 5: Frame assignment in the right front leg

The above figures depict the frames assigned in the right front leg. The DH table similarly needs to be created based on the position and orientation of each frame with respect to the other.

The origin o_0^0 in this case as in the relative position of the first frame is [-0.1016 0.061714 -0.07874] and the orientation RPY is 1.5708 -0.063777 3.1416]. Therefore, based on this pose the origin of frame 1 should be obtained.

Similarly based on the relative positions and orientations of the subsequent frames the DH table is obtained.

Table 6: DH table of right front leg

i	*α*	*a*	θ	*d*
1	-93deg	0	θ1	0.0421
2	145.47deg	-0.127	0	0
3	-25.49 deg	0	θ2	0
4	0	-0.0895	0	0.0149

The transformation matrices are obtained by substituting these values in equation 4 to obtain the end effector coordinates.

7.4 Right Rear Leg

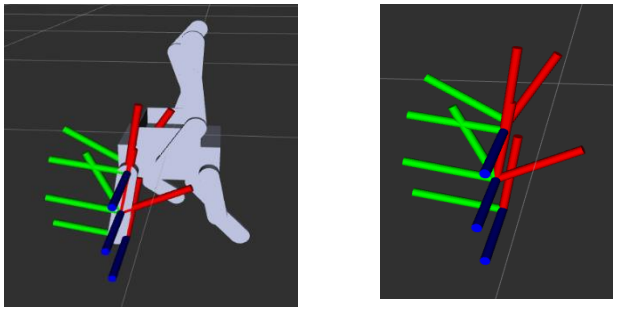

Figure 6: Frame assignment in the right rear leg

The above figures depict the frames assigned in the right front leg. The DH table similarly needs to be created based on the position and orientation of each frame with respect to the other

The origin o_0^0 in this case as in the relative position of the first frame is [0.1016 0.0550 -0.07874] and the orientation RPY is [1.5707963267949 -1.01459157286801 3.14159]. Therefore, based on this pose the origin of frame 1 should be obtained.

Similarly based on the relative positions and orientations of the subsequent frames the DH table is obtained.

Table 7: DH table of right rear leg

i	*α*	*a*	θ	*d*
1	25.74deg	0	π + θ1	0.05
2	-55.85 deg	-0.127	Pi	0
3	69.04 deg	0	π + θ2	0
4	0	-0.0895	0	-0.0161

The transformation matrices are obtained by substituting these values in equation 4 to obtain the end effector coordinates.

The θ1 and θ2 values are modified as in the joint state publisher GUI.

Figure 7: Join State Publisher GUI

8. Inverse Kinematics

Since the leg consists of 2 links. there would be two solutions based on the inverse kinematics problem. Px refers to the projection of the end effector frame in the x direction. Py refers to the projection in the y direction. Based on this the triangles are formed between the different frames for the two solutions.

Elbow down approach

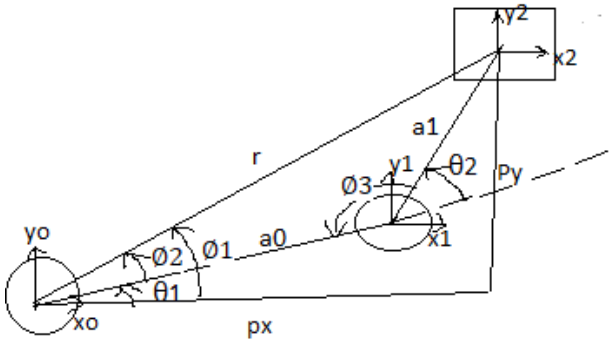

Figure 8: Elbow down approach

$$\emptyset 1 = \tan^{-1}\left(\frac{Py}{Px}\right)$$

$$r = \sqrt{Px^2 + Py^2}$$

$$\emptyset 2 = \cos^{-1}((r^2 + a0^2 - a1^2)/2a0r)$$

theta1 = Ø1 - Ø2

$$\emptyset 3 = \cos^{-1}((a0^2 + a1^2 - r^2)/2a0a1$$

θ2 = π - Ø3

Elbow up approach

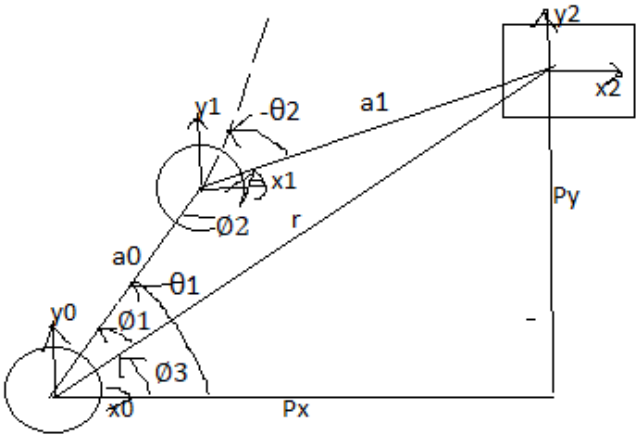

Figure 9: Elbow up approach

$$\emptyset 3 = \tan^{-1}(Py/Px)$$
$$r = \sqrt{Px^2 + Py^2}$$
$$\emptyset 1 = \cos^{-1}((r^2 + a0^2 - a1^2)/2a0r)$$
$$\theta 1 = \emptyset 1 + \emptyset 3$$
$$\emptyset 2 = \cos^{-1}((a0^2 + a1^2 - r^2)/2a0a1$$
$$\theta 2 = \emptyset 2 - \pi$$

Therefore, based on the orientations of the frames in each leg, and substituting the values accordingly will result in the theta1 and theta2 values.

9. Gait Analysis

Gait refers to a pattern of limb actions that a human/animal uses repetitively during locomotion. Among different animals and within an individual animal over time, one can expect to see a variety of locomotion patterns. The different gait patterns in animals are walk, trot, canter, gallop to name a few [9]. The same gait patterns can be implemented for a quadruped robot.

In a walk gait pattern, the robot walks in the 1-3-2-4 order (1: left front leg, 2: right front leg, 3: right rear leg, and 4: left rear leg) and there are three legs on the ground which are in stand phase at any time.

In a trot gait the left front leg and the right rear leg are in support phase when the right front leg and the left rear leg are in swing phase. Similarly, when the right front leg and the left rear leg are in the support phase the left front leg and the right rear leg are in the swing phase.

In the canter gait, one of the robot's rear legs – the right rear leg propels the robot forward. During this phase, the robot is supported only on that single leg while the remaining three legs are moving forward. On the next phase the robot catches itself on the left rear and right front legs while the other rear leg is still momentarily on the ground. On the third phase, the robot catches itself on the left front leg while the diagonal pair is momentarily still in contact with the ground.

In the gallop gait, like a canter, the robot will strike off with its non-leading rear foot; but the second stage of the canter becomes, in the gallop, the second and third stages because the inside rear foot hits the ground a split second before the outside front foot. Then both gaits end with the striking off of the leading leg, followed by a moment of suspension when all four feet are off the ground.

Gait cycle in humans

The gait cycle in humans [8] can be broken down into two primary phases, the stance and swing phases, which alternate for each lower limb.

- Stance phase: Consists of the entire time that a foot is on the ground.
- Swing phase: Consists of the entire time that the foot is in the air

The below diagram shows the different phases in the gait cycle of humans.

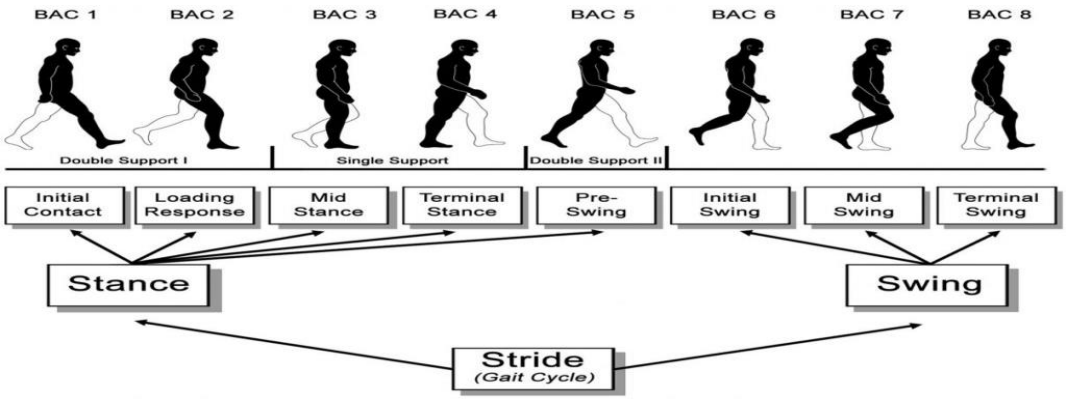

Figure 10: Human Gait Cycle

The gait patterns implemented as part of this project is a walk gait where we will be passing joint angle to each limb of a robot one at a time and a trot g ait where we will be passing joint angles to the diagonal legs at the same time.

10. Software Tools

We make use of SolidWorks to create the CAD model of the robot. The relationships between the different links are established and exported to create a ROS package. We use RVIZ to simulate the forward and inverse kinematics. The ROS package is imported in RVIZ and with the help of a joint state publisher GUI we performed forward kinematics to obtain end effector positions for different joint angles. For the inverse kinematics we made use of ROS publisher and subscriber nodes to obtain joint angles for different end effector positions. To simulate the different gait patterns, we imported the model in Gazebo where we were able to visualize the walking motion of the robot for different gaits.

11. Validation

11.1 FK Validation

The forward kinematics can be validated by comparing the end effector positions which are obtained for different joint angles which can be modified through the joint state publisher with a python script which takes in the joint parameters based on the RVIZ world frame by forming the DH table and computing the transformation matrices based on which the end effector position is obtained.

FK for left front leg

The dummy link is the frame of the robot which is present at the foot. The coordinates of this position would represent the end effector location.

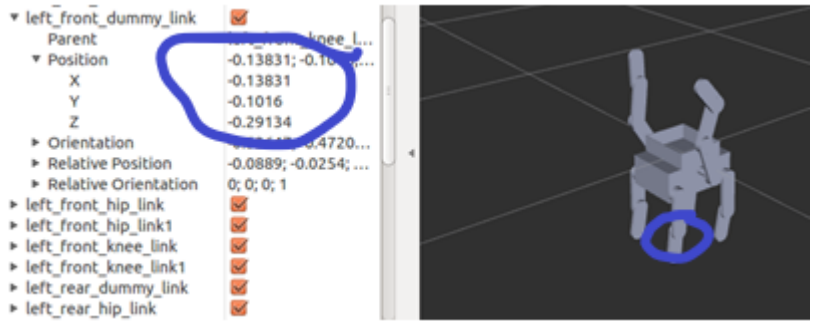

Figure 11: End effector position in RVIZ for right front leg

The end effector coordinates when theta1 and theta2 equal to zero are [-0.138, -0.1016, -0.291].

The python code snippet below will calculate the corresponding theoretical end effector coordinates based on the relative positions and orientations of the different frames.

```
dh = {alpha0: -2.99, a0: 0, d1: 0.01491, q1: 3.14159+(theta1),
      alpha1: 2.71814, a1: -0.127, d2: 0, q2:    3.14159,
      alpha2: 2.6234, a2: 0, d3: 0, q3:   -3.14159+(theta2),
      alpha3: 0, a3: -0.0895, d4: 0.0022, q4: 0
      }
TF1= Matrix
 [[ cos(q), -sin(q)*cos(alpha), sin(q)*sin(alpha), -0.1016 +
a*cos(q)],
  [ sin(q), cos(q)*cos(alpha), -cos(q)*sin(alpha), -0.0889 +
d*0.986+ a*sin(q)],
  [ 0, sin(alpha), cos(alpha), -0.07874 - d*0.161],
  [ 0, 0, 0, 1]]

TF = 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]])
```

The end effector coordinates computed are [-0.1391, -0.074, -0.083].

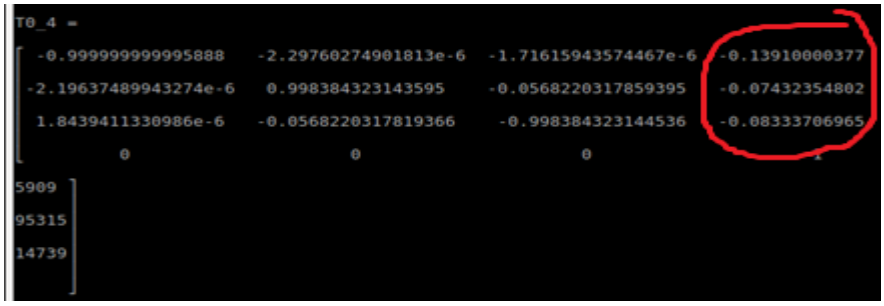

Figure 12: End effector position computed through python script

Similarly, when theta1 = -39.5 degrees, theta 2= 88.23 degrees. This is the initial position in which the robot would stand. The end effector positions obtained in RVIZ are [-0.10983, -0.1016, -0.2079]. The coordinates obtained by computed through python are [-0.182, 0.024, -0.167].

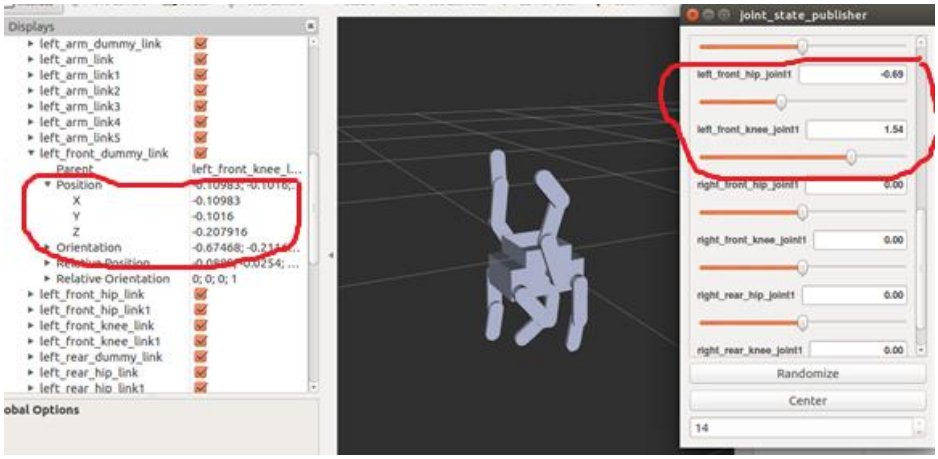

Figure 13: End effector position in RVIZ for different joint angles

```
T0_4 =

[-0.215257347018198  -0.363888890093748  -0.906227979164751  -0.18227634131104

 -0.212445066528862  0.923206748866633   -0.320244270135549  0.0249508649417227

 0.953169118395175   0.123588731336814   -0.276033434975407  -0.167056168346255

 [        0                  0                  0                  1
```

Figure 14: End effector position computed through python script for new joint angles

FK for left rear leg

The end effector coordinates when theta1 and theta2 equal to zero are [0.063703, -0.1016, -0.28851].

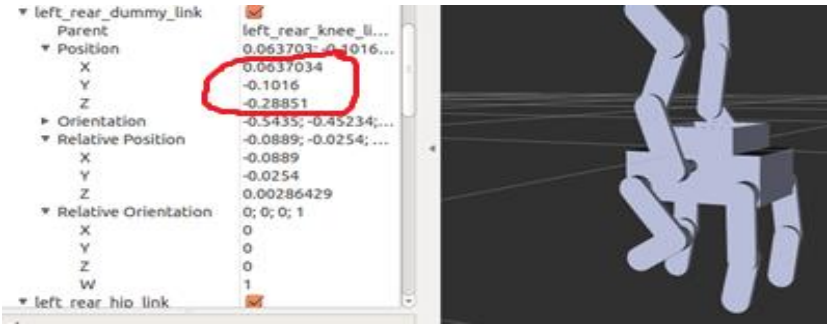

Figure 15: End effector position in RVIZ for left rear leg

This piece of code below will calculate the corresponding theoretical end effector coordinates based on the relative positions and orientations of the different frames.

```
dh = {alpha0: 2.92, a0: 0, d1: 0.015, q1: -3.14159+(theta1),
      alpha1: 1.229, a1: -0.127, d2: 0, q2: 3.14159,
      alpha2: 1.02, a2: 0, d3: 0, q3: 3.14159+(theta2),
      alpha3: 0, a3: -0.0889, d4: 0.0028, q4: 0}
TF1 = Matrix (
[[ cos(q), -sin(q)*cos(alpha), sin(q)*sin(alpha), 0.1016 +
a*cos(q)],
```

```
 [ sin(q), cos(q)*cos(alpha), -cos(q)*sin(alpha), -0.0889 +
d*0.97 + a*sin(q)],
 [ 0, sin(alpha), cos(alpha), -0.07874 + d*0.24],
 [0, 0, 0, 1]])

   TF = 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]])
```

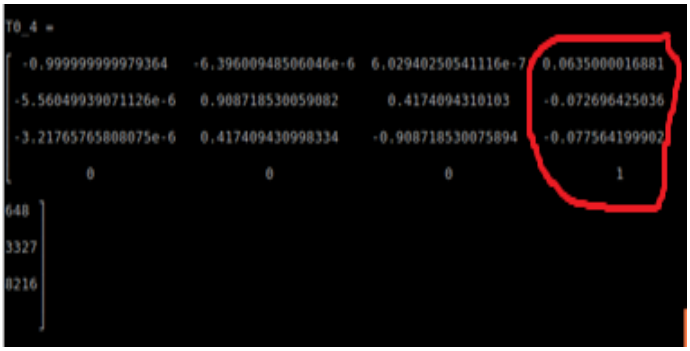

Figure 16: End effector position computed

Similarly, when theta1 = -39.5 degrees, theta 2= 88.23 degrees. This is the initial position in which the robot would stand. The end effector positions obtained in RVIZ are [0.0937, -0.1016, -0.1896]. The end effector position computed through the python script is [-0.042, -0.035, -0.1375].

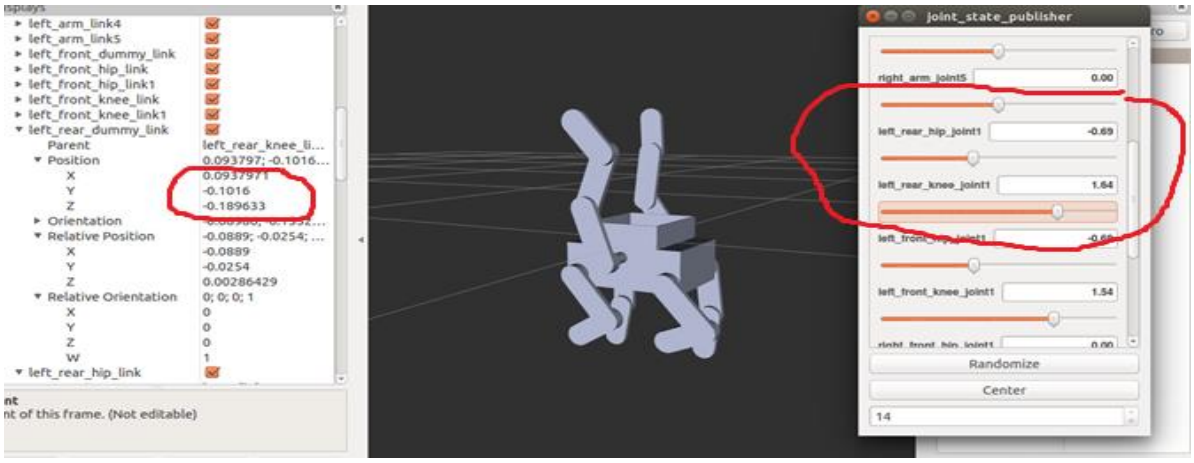

Figure 17: End effector position in RVIZ for right front leg

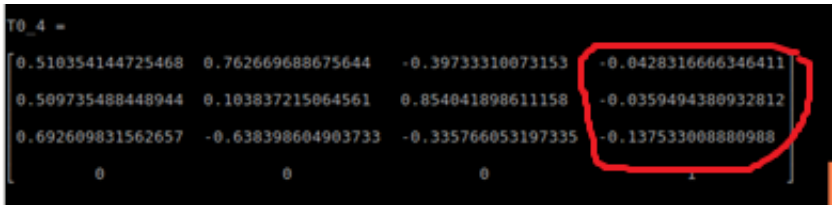

Figure 18: End effector position computed

FK for Right Front leg

When theta1 = -39.5 degrees, theta 2= 93.39 degrees. The end effector positions obtained in RVIZ are [-0.112, -0.0888, -0.194046].

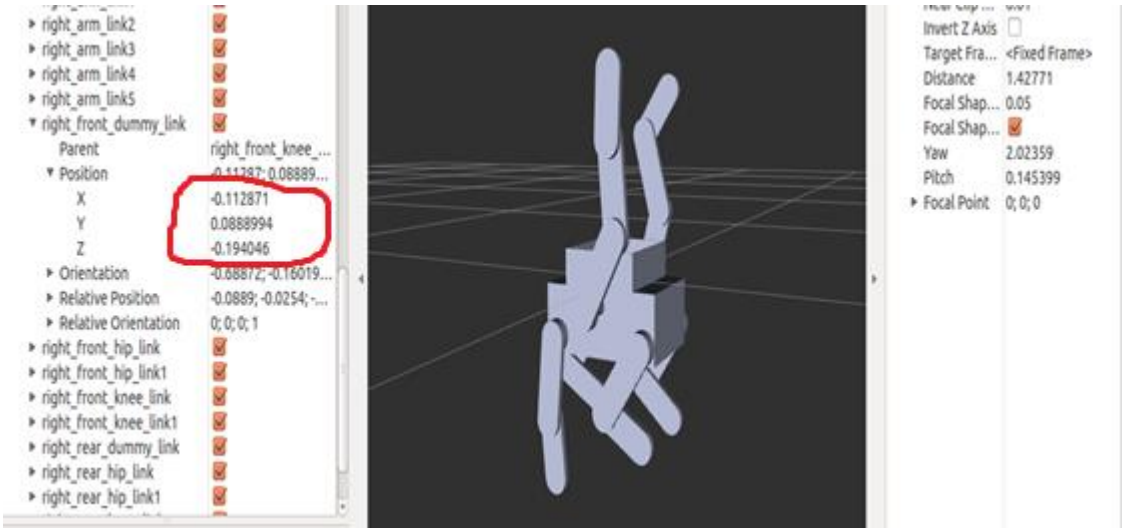

Figure 19: End effector position in RVIZ for right front leg

The piece of code below will calculate the corresponding theoretical end effector coordinates based on the relative positions and orientations of the different frames.

```
dh = {alpha0: -2.313, a0: 0, d1: 0.0421, q1: -0.69,
      alpha1: -2.539, a1: -0.127, d2: 0, q2: 0,
      alpha2: 1.189, a2: 0, d3: 0, q3: 1.64,
      alpha3: 0, a3: -0.0889, d4: 0.0149, q4: 0
      }

TF1 = Matrix (
[[ cos(q), -sin(q)*cos(alpha), sin(q)*sin(alpha), 0.1016 +
a*cos(q)],
 [ sin(q), cos(q)*cos(alpha), -cos(q)*sin(alpha), 0.061714 +
d*0.0622 + a*sin(q)],
 [ 0, sin(alpha), cos(alpha), -0.07874-d*0.998],
 [ 0, 0, 0, 1]])

TF = 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]])
```

The End effector position computed is [-0.195, 0.1188, -0.206]

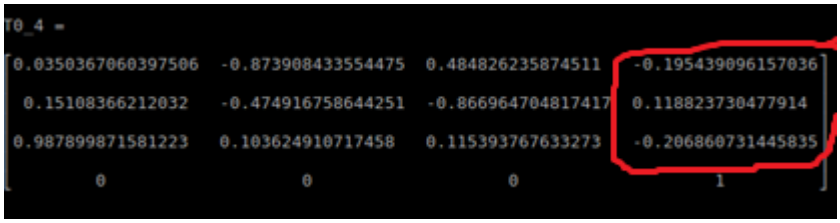

Figure 20: End effector position computed

FK for Right Rear leg

When theta1 = -40.1 degrees, theta 2= 99.7 degrees. The end effector positions obtained in RVIZ are [0.102, 0.0889, -0.1784].

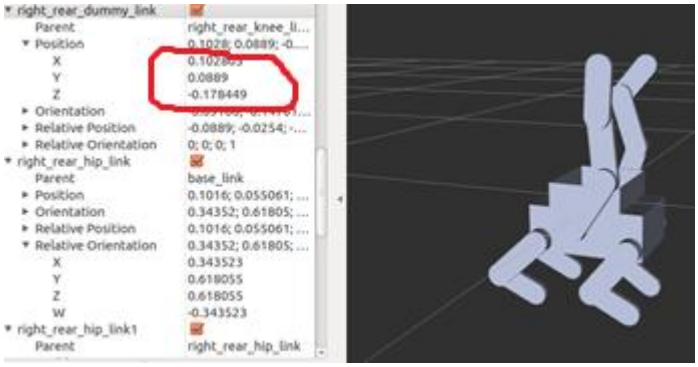

Figure 21: End effector position in RVIZ for right rear leg

```
dh = {alpha0: -0.262, a0: 0, d1: 0.05, q1: theta1,
      alpha1: -0.974, a1: -0.127, d2: 0, q2:     0,
      alpha2: 2.947, a2: 0, d3: 0, q3: theta2,
      alpha3: 0, a3: -0.0889, d4: -0.0161, q4: 0
      }

TF1 = Matrix (
[[ cos(q), -sin(q)*cos(alpha), sin(q)*sin(alpha), 0.1016 +
a*cos(q)],
 [ sin(q), cos(q)*cos(alpha), -cos(q)*sin(alpha), 0.05506 +
d*0.84 + a*sin(q)],
 [ 0, sin(alpha), cos(alpha), -0.07874-d*0.52],
 [ 0, 0, 0, 1]])

TF = 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]])
```

The End effector position computed is [0.0051, 0.1614, -0.0162]

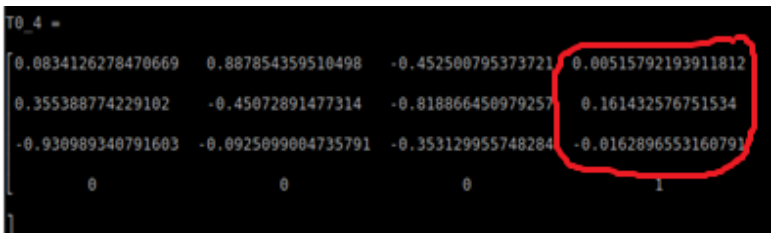

Figure 21: End effector position computed

11.2 *IK validation*

The inverse kinematics of each leg can be validated by comparing the joint angles based on which the end effector positions were observed in RVIZ and using the end effector position to compute the joint angles theta 1 and theta2 through a python script.

We created a publisher script to publish the pose of the end effector positions and a subscriber node to take in these parameters and calculate the joint angles.

The below is code snippet to publish the pose.

```
p.position.x = -0.1381
p.position.y = -0.1016
p.position.z = -0.29134
p.orientation.x = -0.5264
p.orientation.y = -0.4720
p.orientation.z = -0.4720
p.orientation.w = 0.5264
pub.publish(p)
```

This piece of code will subscribe to this pose and compute the joint angles.

```
rospy.Subscriber('/pose', Pose, callback)
a0=-0.127
a1=-0.0895
phi1 = math.degrees(atan2(py,px))
r = sqrt(pow(px,2) + pow(py,2))
phi2 = math.degrees(acos(add=-(a1*a1)+(a0*a0)+(r*r)/(2*a0*r)
))
theta1 = phi1 - phi2
phi3 = math.degrees(acos(-(r*r)+(a1*a1)+(a0*a0)/(2*a0*a1)))
theta2 = math.pi - phi3
theta1=math.radians(theta1)
theta2=math.radians(theta2)
```

IK for left front leg

The end effector positions obtained when theta1 and theta2 are equal to zero are [0.138, -0.1016, -0.291] and orientation [-0.526, -0.472, -0.472,0.526].

We pass this pose in the publisher script and run the subscriber node to obtain the theta1 and theta2 values.

Published pose

```
position:
  x: -0.1381
  y: -0.1016
  z: -0.29134
orientation:
  x: -0.5264
  y: -0.472
  z: -0.472
  w: 0.5264
```

Obtained Joint angles

```
[INFO] [1607537120.866728]: theta1 0.0
[INFO] [1607537120.867195]: theta2 1.47
877932477e-06
```

Figure 22: Joint angles computed for published pose

The end effector pose obtained when theta1 equal to -39.9 degrees and theta2 equal to 93.9 degrees was [-0.10983, -0.1016, -0.2079]. Passing this pose to the subscriber node we obtain theta1 to be -100 radians and theta2 equal to -94 degrees.

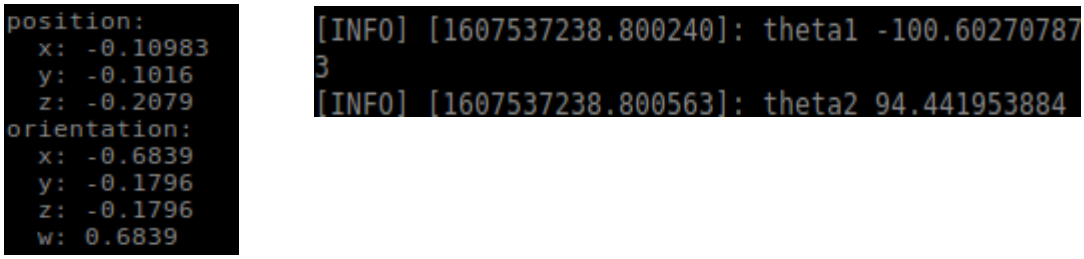

Figure 23: Joint angles computed for published pose

IK for left rear leg

For the left rear leg, the end effector positions obtained for theta1 and theta2 equal zero are [0.063, -0.1016, -0.2885]. Passing this value in the publisher script we obtained theta1 equal to zero and theta2 to be close to zero.

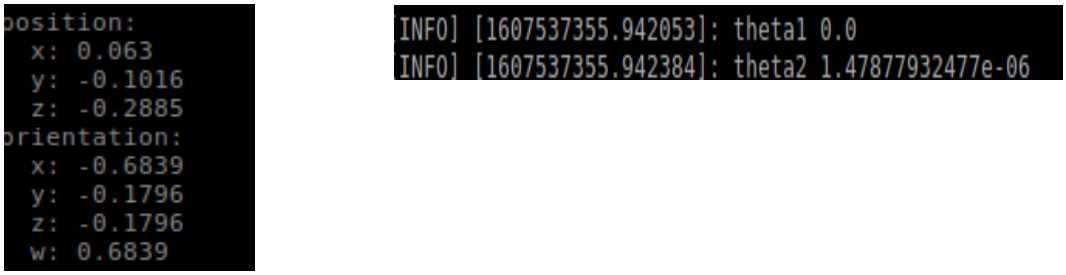

Figure 24: Joint angles computed for published pose

The end effector pose obtained when theta1 equal to -39.9 degrees and theta2 equal to 93.9 degrees was [0.093, -0.1016, -0.1896]. Passing this pose to the subscriber node we obtain theta1 to be -188,275 degrees and theta2 equal to 103.139 degrees.

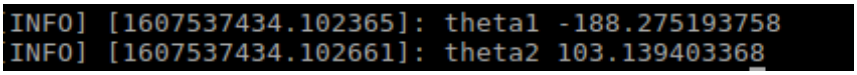

Figure 25: Joint angles computed for published pose

IK for right front leg

The end effector pose obtained when theta1 equal to -40.1 degrees and theta2 equal to 99.8 degrees was [-0.112, 0.088, -0.194]. Passing this pose to the subscriber node we obtain theta1 to be -179.8 degrees and theta2 equal to 99.753 degrees.

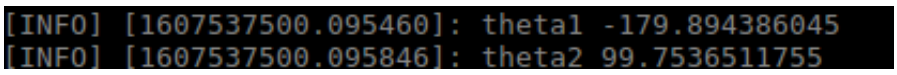

Figure 26: Joint angles computed for published pose

IK for right rear leg

The end effector pose obtained when theta1 equal to -40.1 degrees and theta2 equal to 99.8 degrees was [0.102, 0.0889, -0.1784]. Passing this pose to the subscriber node we obtain theta1 to be -99.1 degrees and theta2 equal to 104.8.

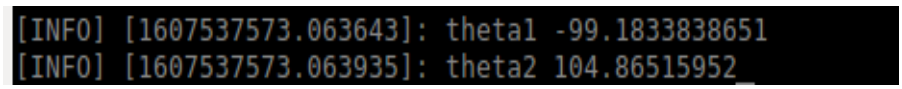

Figure 27: Joint angles computed for published pose

11.3 Gaits Simulation

The simulations for the different gaits were worked on in gazebo. The two gaits we focused on were the walk gait and trot gait. The scripts executing the walk gait sets an initial stance for the robot by publishing joint angles to the hip and knee joints of each leg and the position for the arms to be lifted on top. The subsequent loops execute the forward motion for each leg by publishing joint angles which make the robot move forward. Once all the legs move forward the knee and the hip joint are set back to its original position. This cycle is continuously repeated. The following images will illustrate the different states of the robot as the cycle goes on.

Initial stance

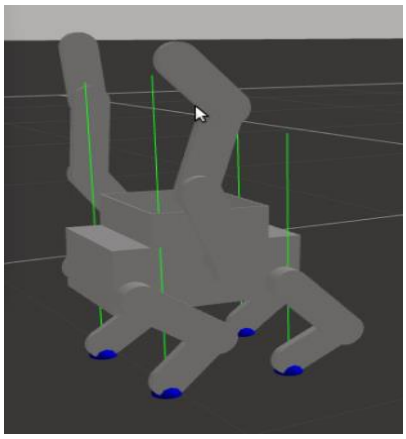

Figure 28: Initial stance

This is the initial position of the robot with each leg having thetha1 (angle at the hip joint) as -39.53 degrees and theta2 (angle at the knee joint) as 93.96 degrees. The angles of the arms were zero degrees. These angles were obtained based on the positions observed in the RVIZ simulation for forward and inverse kinematics. The legs movements can be observed in the gait patterns presented in the following images. The detailed patterns of the leg movements can be viewed in the videos here.

Walk Gait (one leg at a time)

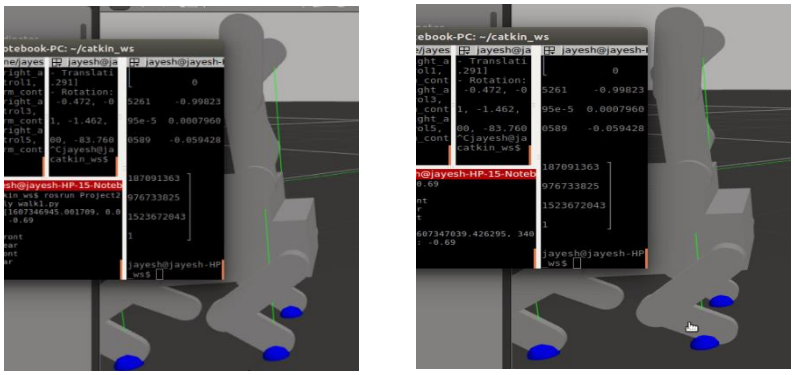

Figure 29: Right Front leg moving forward (before and after movement)

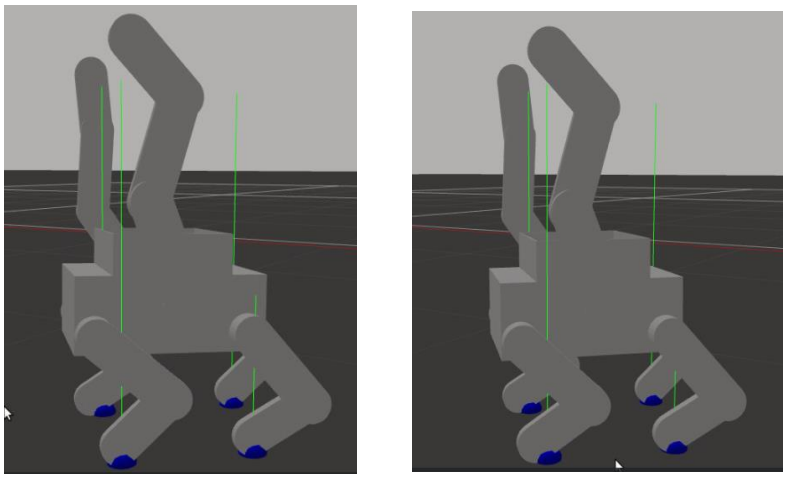

Figure 30: Left Front leg moving forward (before and after movement)

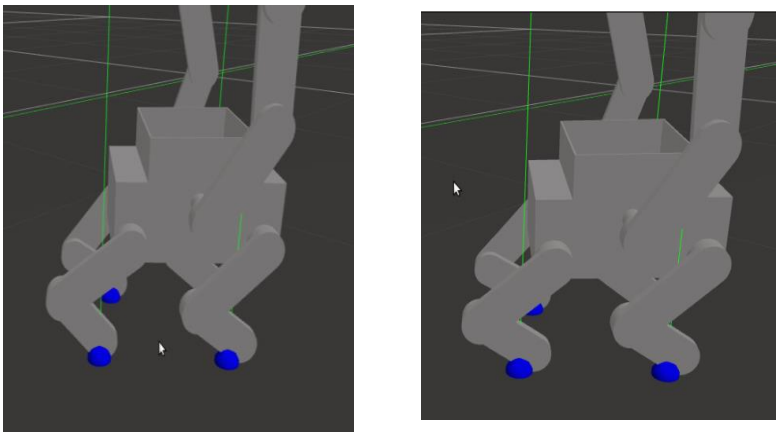

Figure 31: Right rear leg moving forward (before and after movement)

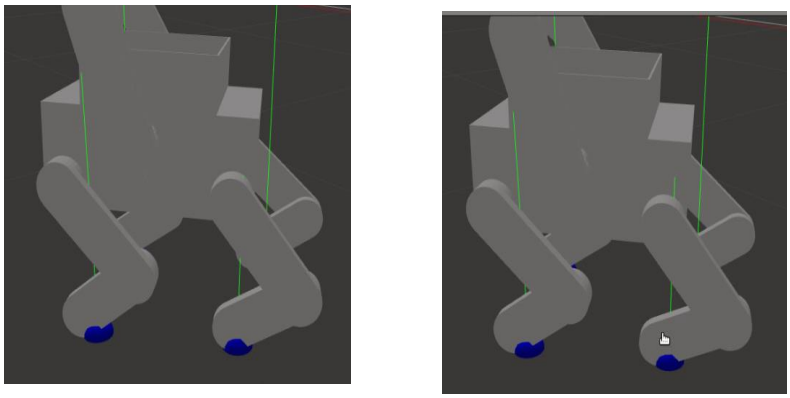

Figure 32: Left rear leg moving forward (before and after movement)

The new angles published at the hip joint for each forward movement were theta1 as -40.7 degrees and theta2 as 99.69 degrees.

Trot gait (Diagonal legs moving at the same time)

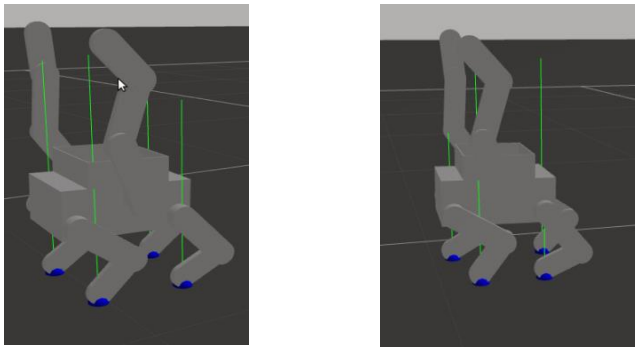

Figure 33: Front right leg & Left Rear leg moving at the same time (initial and final position)

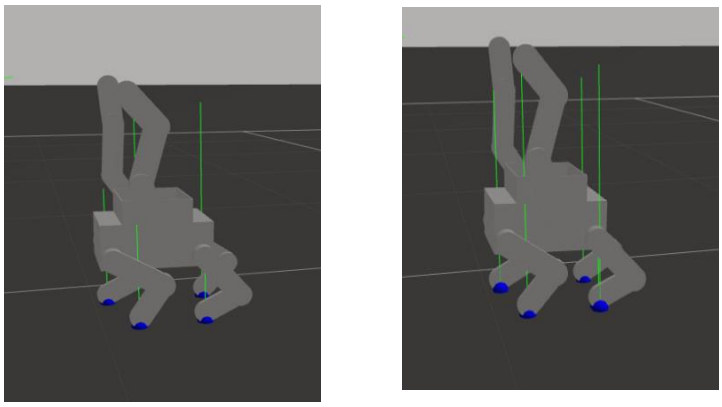

Figure 34: Front Left leg & Right Rear leg moving at the same time (initial and final position)

Even though we did not work on the arm kinematics the below images show the movement of arms to reach a position when passed with different joint angles.

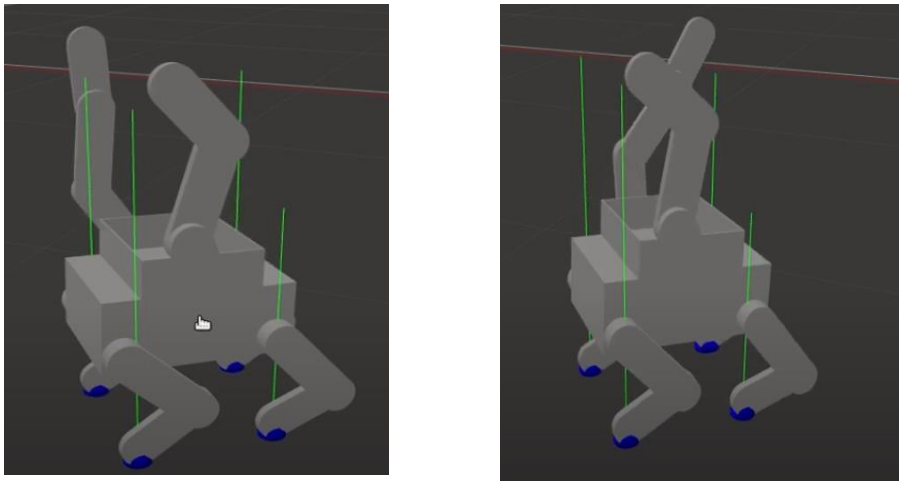

Figure 35: Right arm movement (initial and final position)

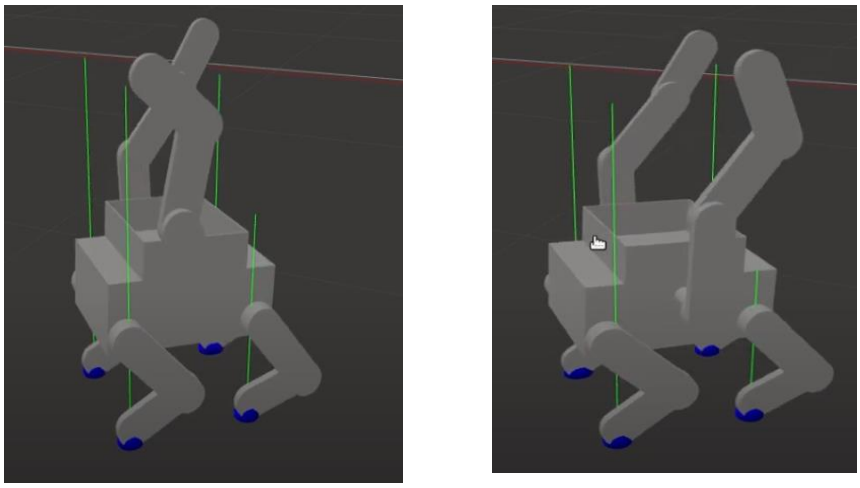

Figure 36: Left arm movement (initial and final position)

12. Discussion and Conclusion

As part of this project, we have developed a model for a quadruped robot with parallel manipulators on each side along with a platform to act as a payload structure to mount objects. The robot was modelled through SolidWorks and was exported as a URDF to create a ROS package based on which it could be launched in the Gazebo and RVIZ world. The forward kinematics of the robot was simulated in RVIZ with the help of a joint publisher GUI where modifying the joint angles resulted in different end effector positions for each leg. The values for the end effector positions obtained in the RVIZ world frame is compared with a python script which computes the end effector positions based on the position and orientation of the different joints. The inverse kinematics is validated by using the end effector positions obtained through FK and publishing it to a python script which computes the corresponding joint angles. The calculations as per the script and the RVIZ coordinates differed in certain instances by small values of up to 0.2 whereas for IK we observed instances when the joint angle theta1 did not match the angles obtained in RVIZ for the different end effector positions but theta 2 was comparable with the values observed in RVIZ. The reason this could not have been the case was because the robot had multiple frames in different positions. The 2-link consisted of 5 frames each oriented differently based on the origin frame. FK and IK for the manipulator were not performed as part of this project

The gait analysis was performed in gazebo with a python script controlling the joints in each leg. The walk and trot gaits were simulated as part of this study. As part of the walk gait each leg of the robot was setup to move one at a time with the stability of the robot maintained. As part of the trot gait the right front leg and the left rear leg were setup to move at the same time and the left front leg and the right rear leg were setup to move at the same time. Both these gait patterns were simulated successfully with the angles of the different joints being hardcoded. Some of the issues faced during the simulation was that all the legs of the robot were not touching the ground with the robot being tilted more on one side. The gravity component in gazebo had to be altered for the robot to have its weight shifted equally on each leg. The arm movements were also simulated by publishing joint parameter values to it. Therefore, based on this project the robot model can be used for various real-world purposes such as delivery, manufacturing, healthcare purposes.

13. Scope for Future Work

Some of the additional tasks which can be carried out to enhance the working model are:

1. The dynamics of the robot can be implemented so that the robot can transfer its weight to different legs accordingly when executing a swing motion.
2. A foot link can be added to the robot by way of a contact sensor where the information can be used to observe the number of legs in contact with the ground to help make leg movements.
3. Grippers can be added to the arm and Arm kinematics can be implemented for the parallel manipulators.
4. We can use gait concepts and inverse kinematics to plan the robot's swing phase and obtain joint angles dynamically based on different end effector positions as the robot moves along.

References

1. https://www.anybotics.com/robotic-package-delivery-with-anymal/
2. https://www.bostondynamics.com/spot/technology#payloads
3. https://www.personalrobots.biz/alphred2-surpass-the-limits-of-quadruped-robots-using-a-clever-design/
4. https://www.bostondynamics.com/atlas
5. Junmin Li, Jinge Wang, Simon X. Yang, Kedong Zhou, Huijuan Tang, "Gait Planning and Stability Control of a Quadruped Robot", Computational Intelligence and Neuroscience, vol. 2016, Article ID 9853070, 13 pages, 2016
6. Tsujita, Katsuyoshi & Tsuchiya, Kazuo & Onat, Ahmet. (2001). Adaptive gait pattern control of a quadruped locomotion robot. 4. 2318 - 2325 vol.4. 10.1109/IROS.2001.976416.
7. O. O. Ortiz, J. Á. Pastor Franco, P. M. Alcover Garau and R. Herrero Martín, "Innovative Mobile Robot Method: Improving the Learning of Programming Languages in Engineering Degrees," in IEEE Transactions on Education, vol. 60, no. 2, pp. 143-148, May 2017, doi: 10.1109/TE.2016.2608779.
8. https://www.protokinetics.com/understanding-phases-of-the-gait-cycle/
9. https://en.wikipedia.org/wiki/Horse_gait
10. https://en.wikipedia.org/wiki/Mobile_robot