REPORT PROJECT-2

ENPM662

MODELING AND SIMULATION OF ROBOT DOG

TEAM MEMBERS

ABDUL NOOR MOHAMMED

USNIK CHAWLA

CONTENTS

INTRODUCTION

Our design is a mechanical robot dog, constructed and simulated for 12 DOF. The robot is modeled with reference measurements in Solid works and then transformed to a URDF file, where it is used to perform simulations in Gazebo and Rviz or any other physics engine. The simulations are certain tasks such as walking or turning. With rapidly increasing DOF, the model becomes increasingly complex to model and run on an engine. However, the Robot dog has proven to be an area of intensive research, possibly because of the endless complex operations it can perform. This robot is a peculiar piece of machine which offers applications ranging from normal animal tasks to increasingly challenging military tasks in harsh terrains. History of the robot dog can be traced back to 'AIBO'; which is a robotic dog designed to be an intelligent and trainable robot companion. Sony unveiled Aibo on 11 May 1999 with a retail price of $2,500. In Japan, the first production series sold out in 20 mins.

With rapid growth in technology, the robot dog has grown powerful. The incorporation of Deep learning and Artificial Intelligence techniques has opened a plethora of applications. Much intelligent robots are deployed to run military operations; they are coupled with advanced image and environment sensing capabilities and can paint a picture for the army. Almost everyone is aware of SPOT; the Boston Dynamics creation, which has become increasingly popular in industries and research facilities.

Figure:1 SPOT from Boston Dynamics

Figure:2 AIBO

APPLICATIONS

This robot offers a multitude of applications. They come with the added advantage of 4 limbs and thus increasing stability and balance for rough terrains. Coupled with sensors and cameras, they become a powerful machine which can be used for industrial and mining sectors.

Some Notable and major applications include:

- Mapping Terrains
- Industrial sensing and reporting
- Underground spot welding
- Mining applications
- Environment sensing and monitoring
- Defensive weapon carriers
- Pet activities
- Security
- Material testing

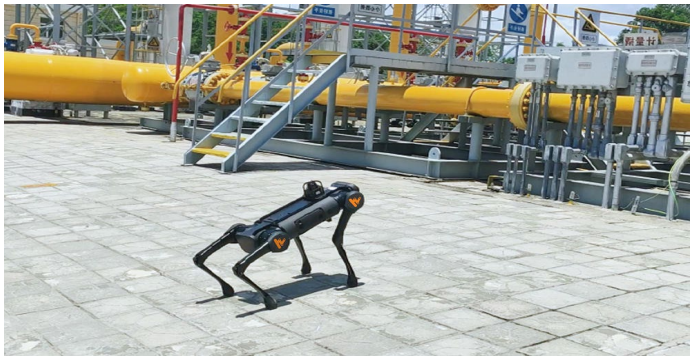


Figure:3 Typical robot applications

GAIT DESIGN

The Quadruped robot consists of a main body and 4 legs for locomotion. The robot is made up of 12 actuators, 3 per leg. The combination of "shoulder" and "elbow" actuators allows the robot to walk linearly. It has 4 legs, identical in geometry.

In order to control a legged robot, one must look to nature and way other legged creatures move. By looking at the movement of quadrupeds in different gaits, determining a control pattern was possible. Quadrupeds have various gaits, both symmetrical, meaning that any given side's (left or right) pair alternate, and asymmetrical, meaning that the pairs move together.

The first gait in the symmetrical gaits section of Figure 4 shows the walking trot. This gait requires the diagonal leg pairs to alternate phases with a short time where all four legs are in contact. The second walking gait is the lateral sequence gait which is more complicated, but allows for potential for three legs on the ground at a given time with worst case being two legs on the ground.

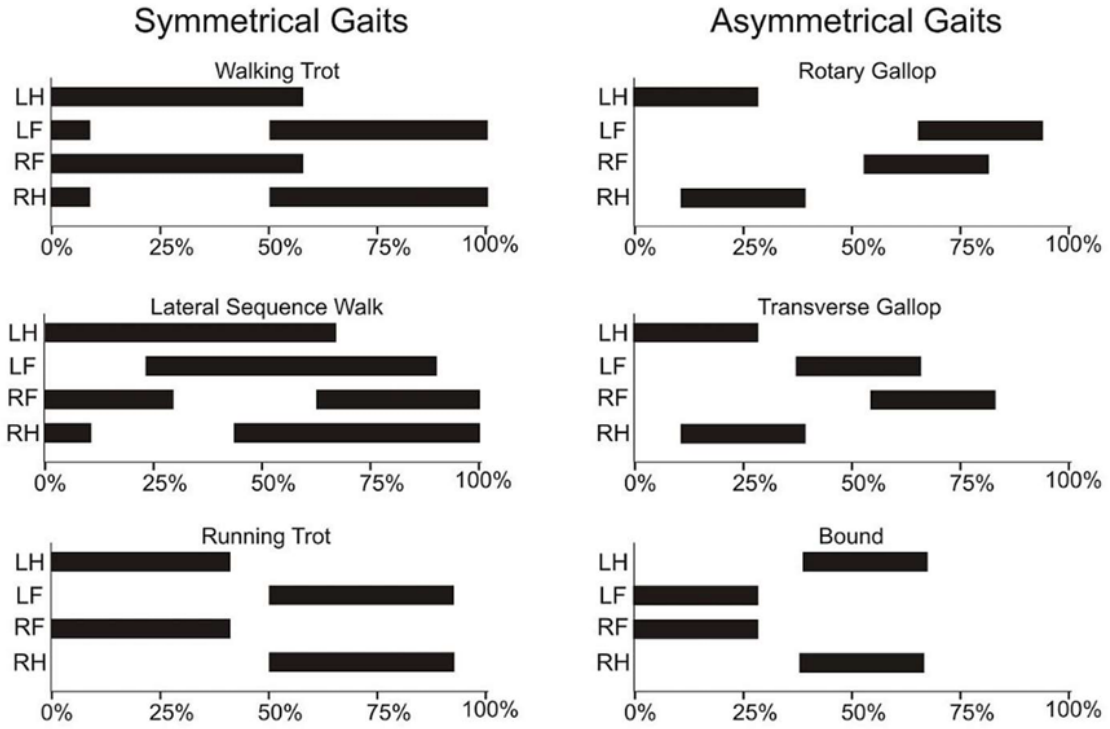


Figure 4: Gait Patterns

DIMENSIONS AND MEASUREMENTS

main_body

Component Instances (1)
Area 1.335E+05 mm^2
Density 0.001 g / mm^3
Mass 387.559 g
Volume 3.663E+05 mm^3
Physical Material (Various)

Bounding Box
Length 160.00 mm
Width 220.00 mm
Height 33.00 mm
World X,Y,Z 0.00 mm, 0.00 mm, 0.00 mm
Center of Mass 0.012 mm, 0.014 mm, 15.60 mm

(Main Body dimensions)

link_shoulder1 v1

Component Instances (1)
Area 9836.228 mm^2
Density 0.001 g / mm^3
Mass 19.857 g
Volume 1.624E+04 mm^3
Physical Material (Various)

Bounding Box
Length 37.00 mm
Width 45.00 mm
Height 17.50 mm
World X,Y,Z 0.00 mm, 0.00 mm, 0.00 mm
Center of Mass -8.165 mm, 22.472 mm, -3.831E-04 mm

(Shoulder Joint dimensions)

link_femur1 v1

Component Instances (1)
Area 1.006E+04 mm^2
Density 0.001 g / mm^3
Mass 21.797 g
Volume 1.807E+04 mm^3
Physical Material (Various)

Bounding Box
Length 29.50 mm
Width 30.00 mm
Height 85.00 mm
World X,Y,Z 0.00 mm, 0.00 mm, 0.00 mm
Center of Mass -12.278 mm, 3.634E-04 mm, -31.262 mm

(Femur or the connecting link dim)

link_forearm1, link_forearm2

Longer_tibia

Component Instances (1)
Area 3859.718 mm^2
Density 0.001 g / mm^3
Mass 7.298 g
Volume 6885.015 mm^3
Physical Material ABS Plastic

Bounding Box
Length 5.00 mm
Width 20.00 mm
Height 92.50 mm
World X,Y,Z -2.50 mm, 0.00 mm, 0.00 mm
Center of Mass -2.50 mm, 0.90 mm, -32.102 mm

(Limb or the final link dim)

CAD MODEL

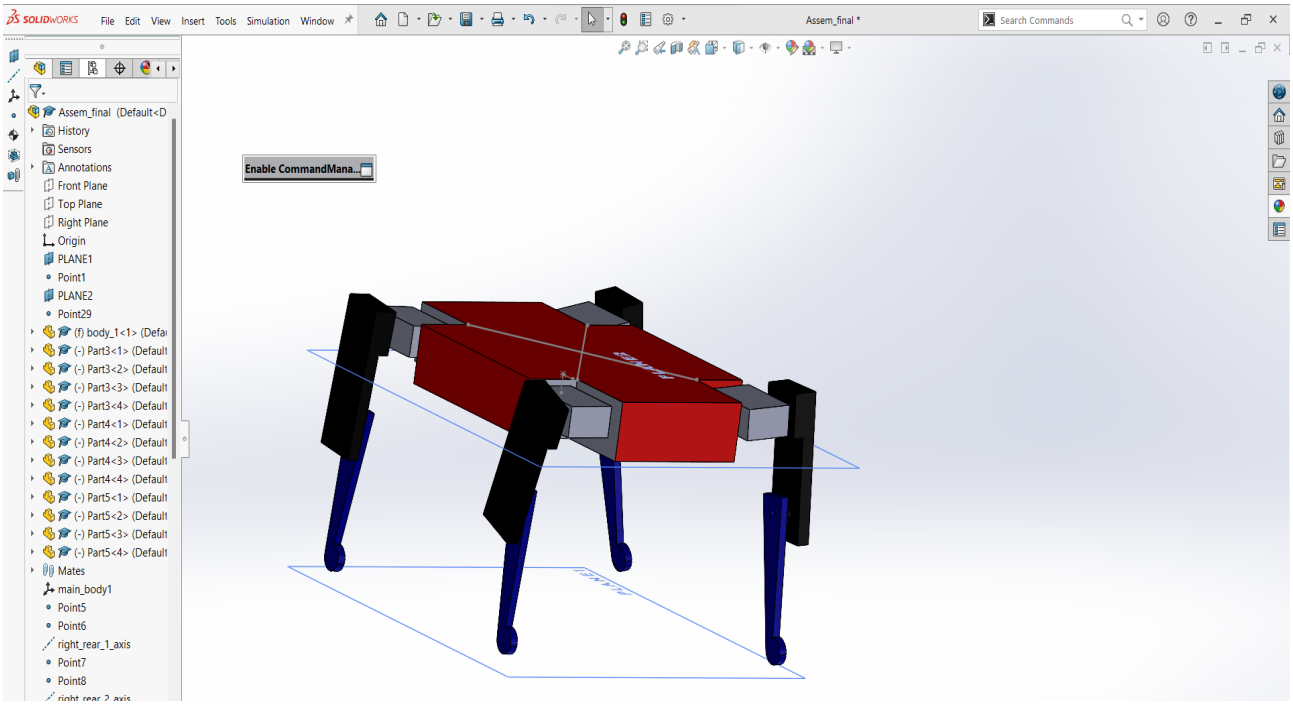
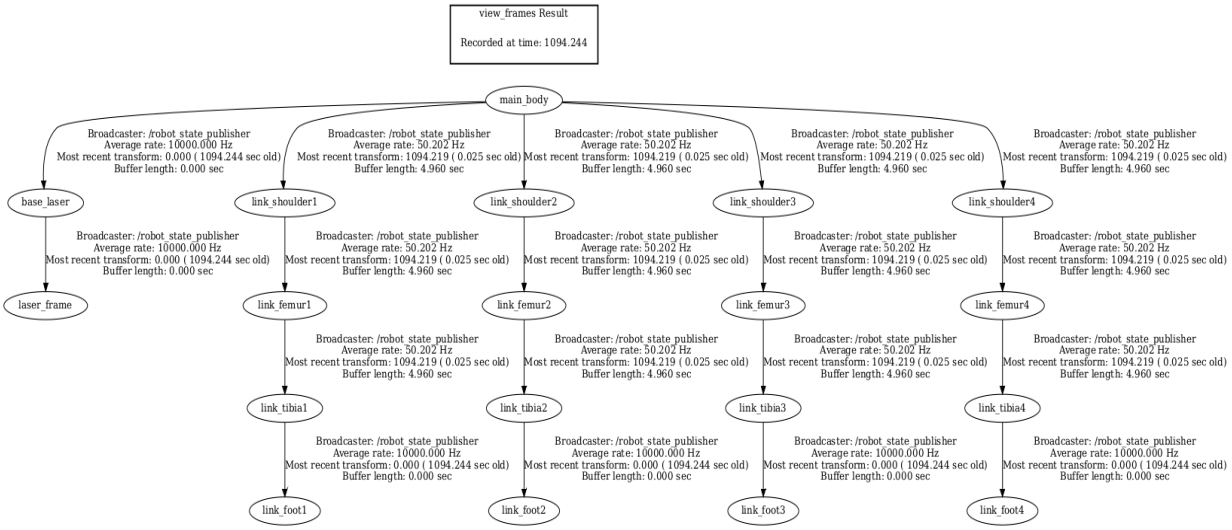


Figure 5: Our CAD model (Solidworks)

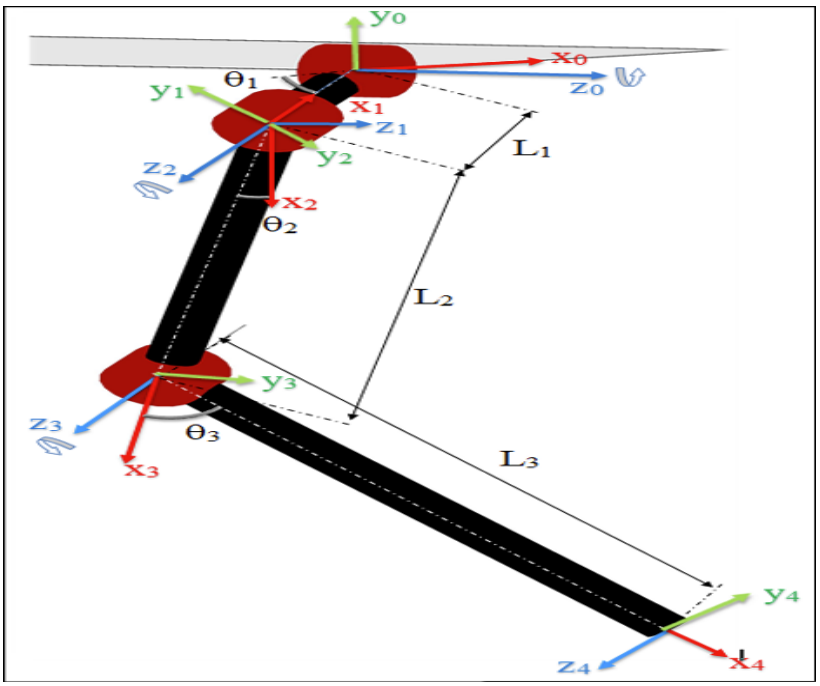
The model is a 12DOF robot. The main body on which the sensors are mounted is the parent for all other links, it is followed by 4 shoulder joints connected to 4 femur links and finally attached to the legs or the limbs. The dimensions are mentioned on the previous page. The dimensioning is a critical aspect and requires a lot of trial-and-error mechanisms. The axes and rotation joints must be accurate otherwise the URDF file will run into errors. The links are attached by a revolute joint with 1DOF.

Tree structure of the Links



DH TABLE

The DH table is simple for the legs of the quadruped and is as shown.



Link	α_{i-1}	a_{i-1}	d_i	θ_i
0-1	0	L1	0	θ1
1-2	-π/2	0	0	-π/2
2-3	0	L2	0	θ2
3-4	0	L3	0	θ3

Figure 6: DH table

FORWARD KINEMATICS

From the above DH Table, the following forward kinematics equations are derived.

$$T_0^1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & -L_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & -L_1\sin(\theta_1) \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & L_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^4 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & L_3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0^4 = T_0^1\, T_1^2\, T_2^3\, T_3^4 = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

$m_{11} = \cos(\theta_2)\cos(\theta_3)\sin(\theta_1) - \sin(\theta_1)\sin(\theta_2)\sin(\theta_3)$
$m_{12} = -\cos(\theta_2)\sin(\theta_1)\sin(\theta_3) - \cos(\theta_3)\sin(\theta_1)\sin(\theta_2)$
$m_{13} = -\cos(\theta_1)$
$m_{14} = L_2\cos(\theta_2)\sin(\theta_1) - L_1\cos(\theta_1) + L_2\cos(\theta_2)\cos(\theta_3)\sin(\theta_1) - L_3\sin(\theta_1)\sin(\theta_2)\sin(\theta_3)$
$m_{21} = \cos(\theta_1)\sin(\theta_2)\sin(\theta_3) - \cos(\theta_1)\cos(\theta_2)\cos(\theta_3)$
$m_{22} = \cos(\theta_1)\cos(\theta_2)\sin(\theta_3) + \cos(\theta_1)\cos(\theta_3)\sin(\theta_2)$
$m_{23} = -\sin(\theta_1)$
$m_{24} = L_3\cos(\theta_1)\sin(\theta_2)\sin(\theta_3) - L_2\cos(\theta_1)\cos(\theta_2) - L_3\cos(\theta_1)\cos(\theta_2)\cos(\theta_3) - L_1\sin(\theta_1)$
$m_{31} = \cos(\theta_2)\sin(\theta_3) + \cos(\theta_3)\sin(\theta_2)$
$m_{32} = \cos(\theta_2)\cos(\theta_3) - \sin(\theta_2)\sin(\theta_3)$
$m_{33} = 0$
$m_{34} = L_2\sin(\theta_2) + L_3\cos(\theta_2)\sin(\theta_3) + L_3\cos(\theta_3)\sin(\theta_2)$
$m_{41} = 0$
$m_{42} = 0$
$m_{43} = 0$
$m_{44} = 1$

The above equations are the derived theroretical kinematic equations. There is a script (forward_kinematics.py) in the package to calculate the forward kinematic equations of the robot leg.

INVERSE KINEMATICS

Calculating the inverse kinematics part is a bit tricky, especially when we have a 12DOF robot and we need specific mathematical manipulations to do those. The above process describes the derivation of Inverse Kinematics.

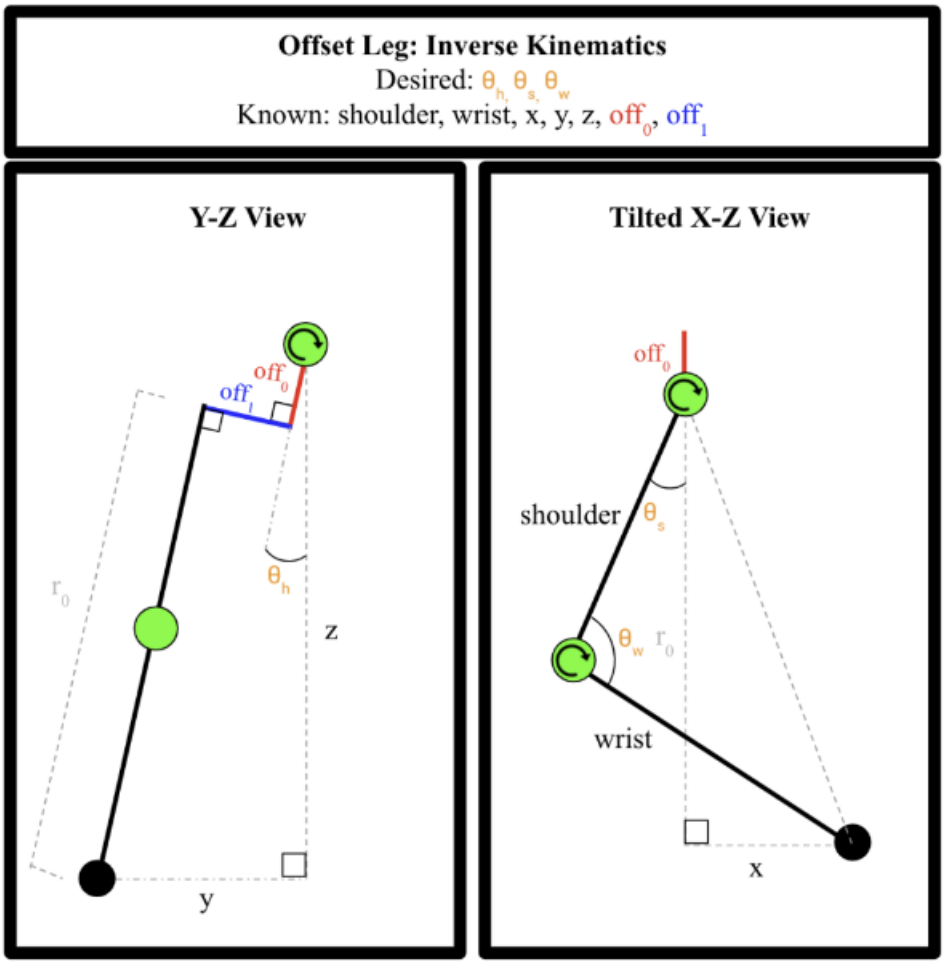
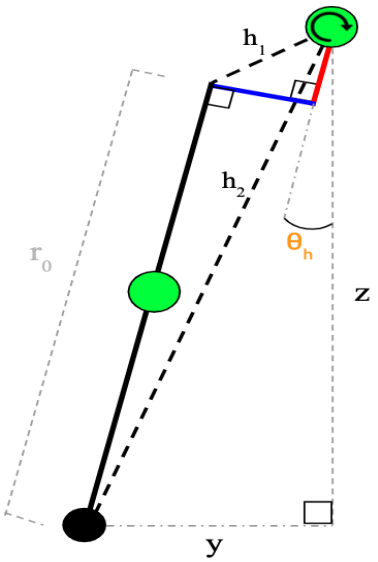
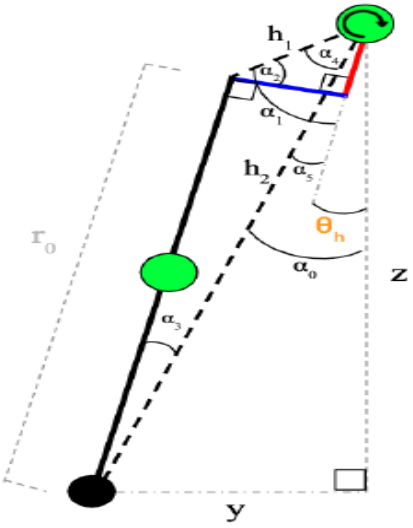


Fig 9: Robot pose for calculating IK



$h_1 = \sqrt{off_0{}^2 + off_1{}^2}$ (ref the first image for offset positions)

$h_2 = \sqrt{z^2 + y^2}$



$\alpha_0 = \arctan\left(\frac{y}{z}\right)$

$\alpha_1 = \arctan\left(\frac{off_1}{off_0}\right)$

$\alpha_2 = \arctan\left(\frac{off_0}{off_1}\right)$

$\alpha_3 = \arcsin\left(\frac{h_1 * \sin(\alpha_2 + 90)}{h_2}\right)$ (law of sines)

$\alpha_4 = 90 - (\alpha_3 + \alpha_2)$

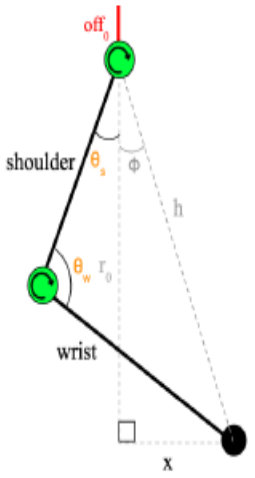
$\alpha_5 = \alpha_1 - \alpha 4$

And finally: $\theta_h = \alpha_0 - \alpha_5$

We can then use the Law of Sines to derive r_0:

$$\frac{r_0}{\sin \alpha_4} = \frac{h_1}{\sin \alpha_3}$$

$$r_0 = \frac{h_1 * \sin \alpha_4}{\sin \alpha_3}$$



We already know: shoulder, wrist, x, and r_0. The first step in this view is to compute the third leg's value (the rightmost leg labeled h):

$$h = \sqrt{r_0^2 + x^2}$$

Now, compute ϕ through simple trigonometry:

$$\sin \phi = \frac{x}{h}$$

$$\phi = \arcsin(\frac{x}{h})$$

All sides of the visible triangle are now explicitly defined. Use Law of Cosines to get the θ_s (shoulder angle):

Note that s & w correspond to shoulder & wrist lengths, respectively.

$$\cos(\theta_s + \phi) = \frac{h^2+s^2-w^2}{2h*s}$$

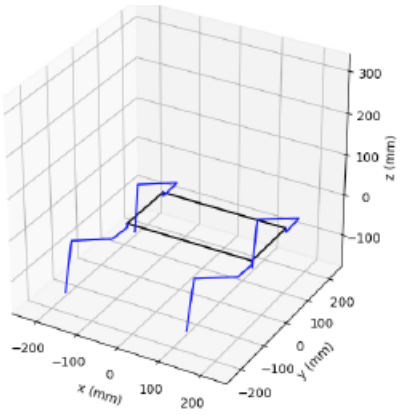
$$\theta_s = \arccos(\frac{h^2+s^2-w^2}{2h*s}) - \phi$$

Now that the θ_s (shoulder angle) is computed, the only value that remains is θ_w (wrist angle) using the Law of Cosines:

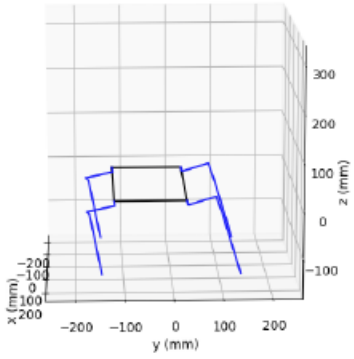
$$\theta_w = \arccos(\frac{w^2+s^2-h^2}{2*w*s})$$

At this point, we now have the inverse kinematics for a 3 DOF leg completely solved. This means that given any xyz coordinate (within the bounds of the control range of the actuator), we can determine the exact joint angles–and thus the motor angles–that we need to go to.

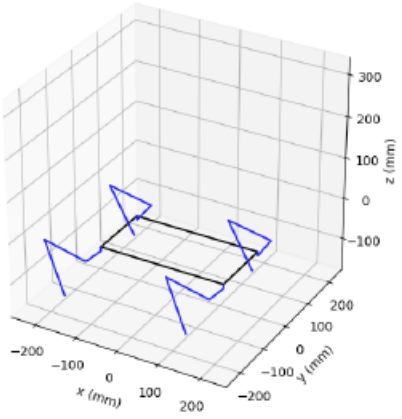
VISUALIZATION



Body X Translation



Body Y Translation



Body Z Translation

Figure 10: Visualization of Kinematics

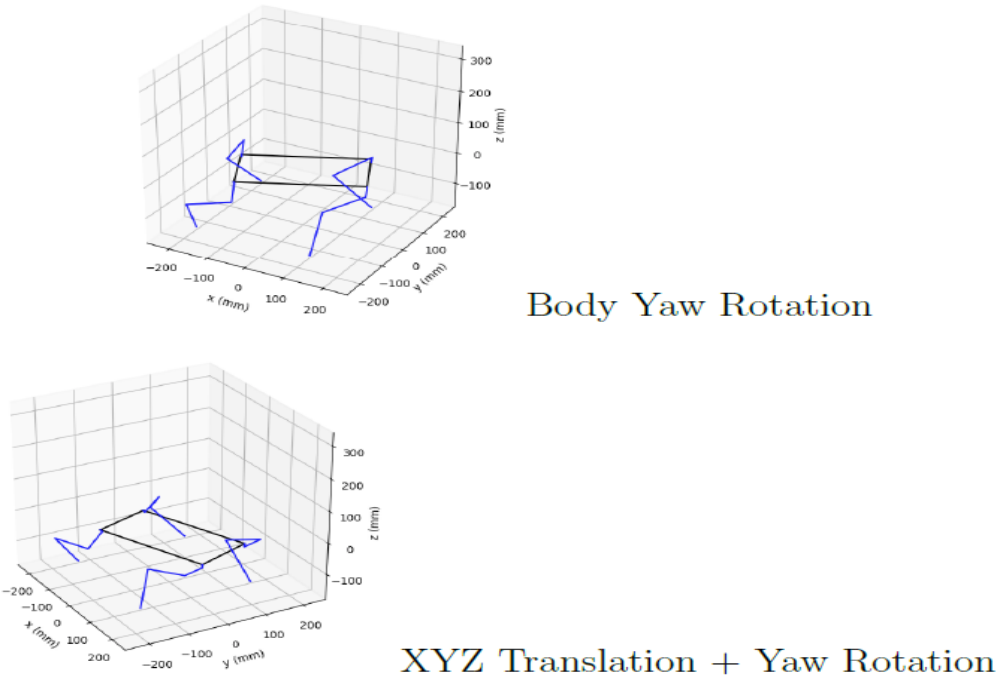


Figure 11: Rotation and translation of robot

WORKSPACE

During the trot gait, RoboDog made short and quick movements of the feet and shuffled forward. An increase in the distance of the feet travel could potentially make this trot gait better, but the vertical height might need to be reduced to increase the useful workspace.

RoboDog's lateral sequence gait was more promising as the quadruped could make more natural looking steps forward. Graphs of these workspaces can be found below in Figure 12

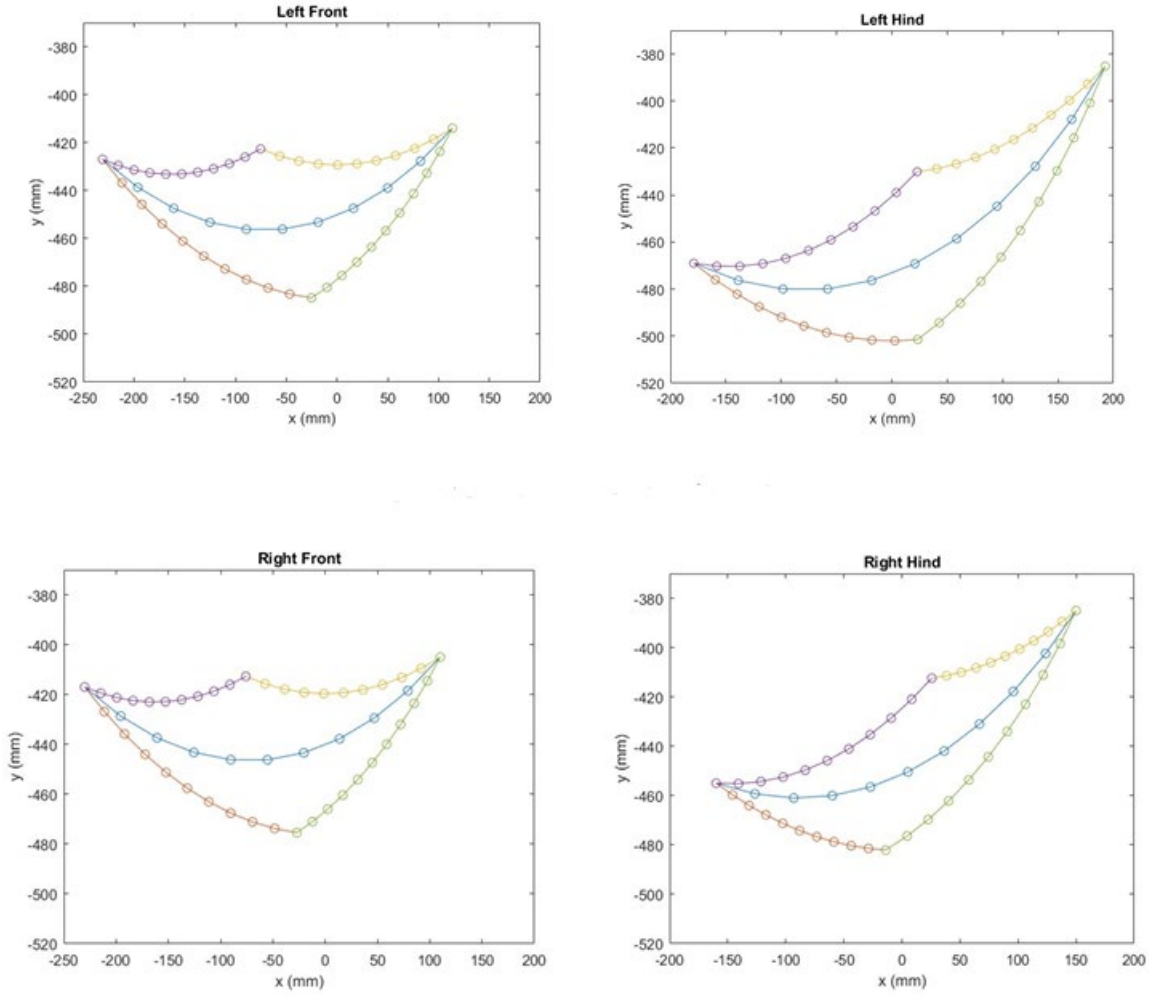


Figure 12: Workspace for the robot dog

ASSUMPTIONS

- The body is assumed to be rigid one with the legs being a series of links.
- We assume no singularity.
- We assume that the joint can perform only one functionality i.e., a joint can be prismatic or revolute and not both.

CONTROL METHOD

We have used joint effort controllers along with a simple PID controller which is defined in the YAML file.

```yaml
# Publish all joint states -----------------------------------
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50

# Position Controllers ---------------------------------------
shoulder_joint1_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_joint1
  pid: {p: 10.0, i: 0.50, d: 0.0}
shoulder_joint2_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_joint2
  pid: {p: 10.0, i: 0.50, d: 0.0}
shoulder_joint3_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_joint3
  pid: {p: 10.0, i: 0.50, d: 0.0}
shoulder_joint4_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_joint4
  pid: {p: 10.0, i: 0.50, d: 0.0}
limb_joint1_position_controller:
  type: effort_controllers/JointPositionController
  joint: limb_joint1
  pid: {p: 10.0, i: 0.00, d: 0.0}
limb_joint2_position_controller:
  type: effort_controllers/JointPositionController
  joint: limb_joint2
  pid: {p: 10.0, i: 0.00, d: 0.0}
limb_joint3_position_controller:
  type: effort_controllers/JointPositionController
  joint: limb_joint3
  pid: {p: 10.0, i: 0.00, d: 0.0}
limb_joint4_position_controller:
  type: effort_controllers/JointPositionController
  joint: limb_joint4
  pid: {p: 10.0, i: 0.00, d: 0.0}
knee_joint1_position_controller:
  type: effort_controllers/JointPositionController
  joint: knee_joint1
  pid: {p: 10.0, i: 0.00, d: 0.0}
knee_joint2_position_controller:
  type: effort_controllers/JointPositionController
  joint: knee_joint2
  pid: {p: 10.0, i: 0.00, d: 0.0}
knee_joint3_position_controller:
  type: effort_controllers/JointPositionController
  joint: knee_joint3
  pid: {p: 10.0, i: 0.00, d: 0.0}
knee_joint4_position_controller:
```

Figure 13: YAML file

GAZEBO SIMULATION

The simulation is performed for a normal Gait, and it was essential to launch it with the right orientation. It was difficult to launch the robot with our desired orientation requiring a massive error resolving.

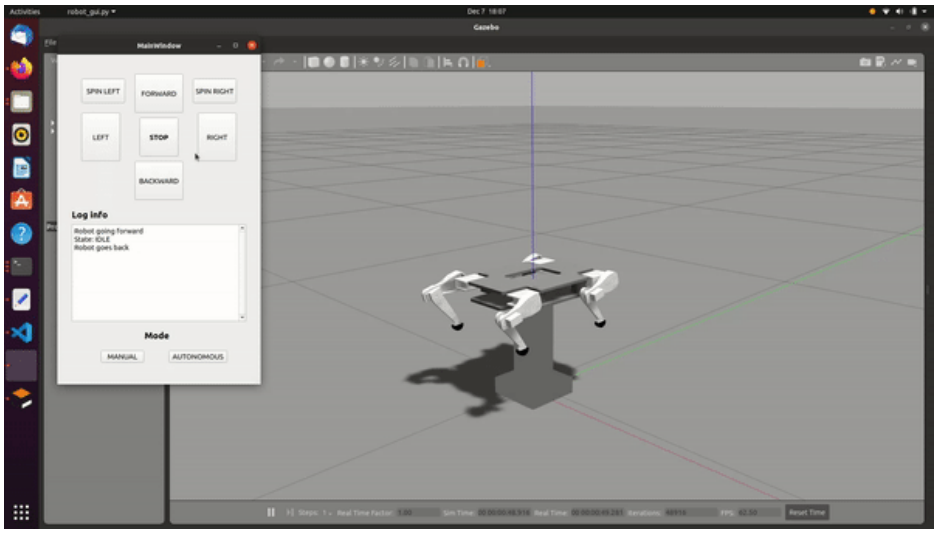


Figure 14: Right- Lateral Movement

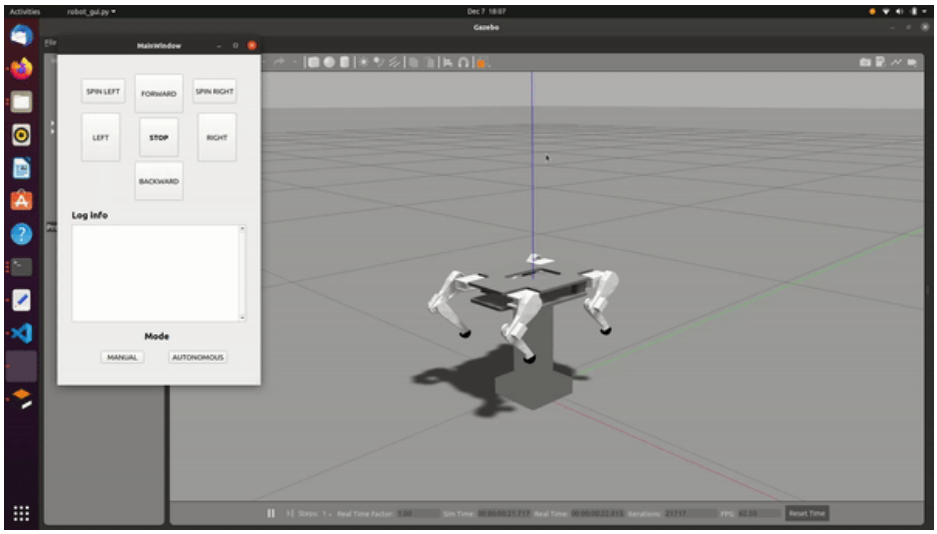


Figure 15: Forward Movement

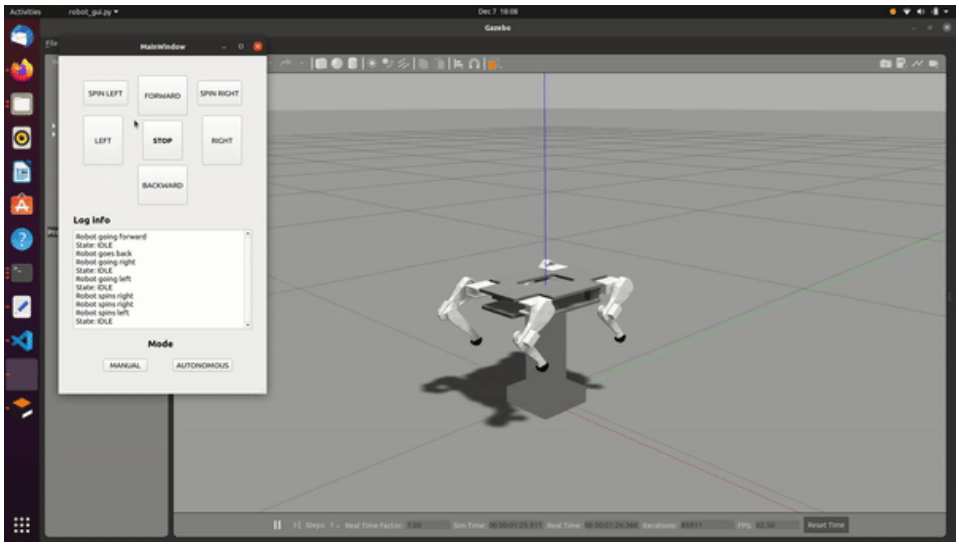


Figure 16: spin left movement

RVIZ OUTPUT

For our robot we have attached a laser scanner on the top of the main body. The output of this scan is visible in the Rviz window as shown:

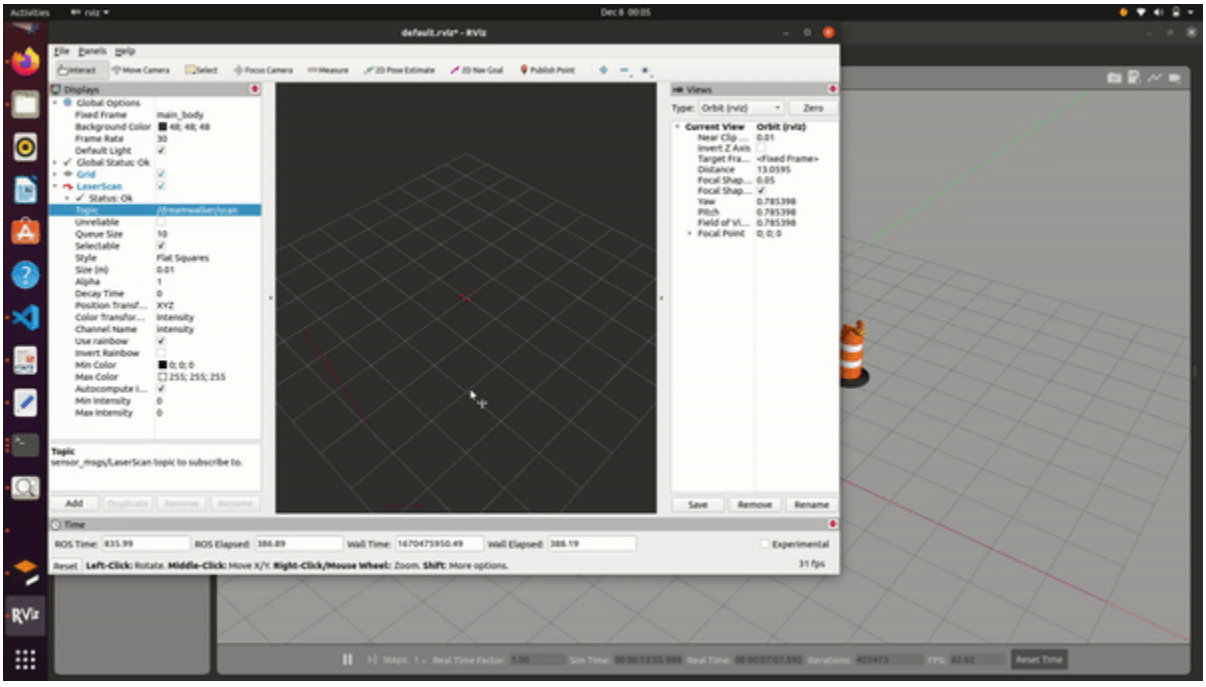


Figure 17: Rviz Visualization

PROBLEMS

- One of the biggest challenges was the URDF conversion and launching in Gazebo. The automated system of selection of axes in Solidworks threw many errors which had to be resolved by manual inputs.
- The robot when launched was not in its desired orientation and several changes had to be made in the URDF scripts to obtain the first initial pose.
- Calculation of friction for the point of contact was tiring and spawning the robot was an even bigger task.
- The robot launches files had to be studied thoroughly from existing files in order to modify and align it with the ground in Gazebo simulation.
- Designing the gui interface was complex and took multiple iterations.

FUTURE WORKS

- It can be further optimized by adding stereo cameras all over the body of the quadruped to estimate the terrain the robot is trying to navigate.
- We can also include dynamic gait generators to respond to sudden changes in terrain like a steep slope or a slippery surface.
- We can also incorporate simultaneous localization and mapping algorithms working in tandem with the laser scanner attached to the robot to perform autonomous navigation.

REFERENCES

- https://github.com/popi-mkx3/popi_project
- 12-DOF Quadrupedal Robot: Inverse Kinematics by Adham Elarabawy
- Gait Analysis of Quadruped Robot Using the Equivalent Mechanism Concept Based on Metamorphosis by Kun Xu, Peijin Zi and Xilun Ding*