|  |  |
| --- | --- |
| Project Work – Group 3  SIMULATION OF A QUADRUPED ROBOT IN ROS | AUTHOR:  Jeen Ann Abraham – S355989,  Kristoffer Anderssen Valderhaug – S325927,  Marius Skaanes – S325939  COURSE WORK: ACIT4820-1 20H APPLIED ROBOTICS AND AUTONOMOUS SYSTEMS |

CONTENTS

[**Chapter 1 : INTRODUCTION** 3](#_Toc57195460)

[1.1. Open Dynamic Robot Initiative 3](#_Toc57195461)

[1.2. Objective 4](#_Toc57195462)

[**Chapter 2 : BACKGROUND** 5](#_Toc57195463)

[2.1. Related Works 5](#_Toc57195466)

[2.2. ROS – Robot Operating System 5](#_Toc57195467)

[2.2.1. ROS Master 5](#_Toc57195468)

[2.2.2. Nodes, Subscribers and Publishers 5](#_Toc57195469)

[2.2.3. Launch Files 5](#_Toc57195470)

[2.2.4. RViz 6](#_Toc57195471)

[**Chapter 3 : 3-DIMENSIONAL MODELLING** 7](#_Toc57195472)

[3.1. Preparing the Mesh 7](#_Toc57195474)

[3.1.1. Tools 7](#_Toc57195475)

[3.2. Quadruped Robot 3-Dimensional Models 7](#_Toc57195476)

[3.3. Physical Parameters 8](#_Toc57195482)

[3.4. MeshLab reports 9](#_Toc57195483)

[3.4.1. Body 9](#_Toc57195493)

[3.4.2. Hip 9](#_Toc57195494)

[3.4.3. Upper leg 10](#_Toc57195495)

[3.4.4. Lower leg 10](#_Toc57195496)

[3.4.5. Foot 11](#_Toc57195497)

[**Chapter 4 : ROS SIMULATION** 12](#_Toc57195498)

[4.1. Introduction 12](#_Toc57195503)

[4.2. Structure of ROS Package. 12](#_Toc57195506)

[4.3. Model Anatomy 13](#_Toc57195507)

[4.4. Simulation of Model in rviz 13](#_Toc57195508)

[4.5. Simulation of Model in Gazebo 14](#_Toc57195509)

[4.6. Challenges in Transition of Quadruped Robot to Champ 15](#_Toc57195510)

[**Chapter 5 : FUTURE WORK** 17](#_Toc57195511)

[5.1. Improvement in model 17](#_Toc57195513)

[5.2. Controller Tuning 17](#_Toc57195514)

[5.3. Implementation of Algorithms for GAIT 17](#_Toc57195515)

[**APPENDIX I: REFERENCES** 19](#_Toc57195516)

[**APPENDIX II: SOURCE CODE** 20](#_Toc57195517)

List of Figures

Figure 1: Open Dynamic Robot Initiative 3

Figure 2: RViz Window Running Test Map 6

Figure 3: Body of quadruped robot 7

Figure 4: Hips 8

Figure 5: Upper leg 8

Figure 6: Lower leg 8

Figure 7: Foot 8

Figure 8: rviz sinulation results for quadruped robot 13

Figure 9: rqt\_gui window 14

Figure 10: Gazebo simulation results for quadruped robot 15

Figure 11: Gazebo running in test world with obstacles 15

Figure 12: RQT Graph showing nodes and topic connections 16

Figure 13:Camera feed from gazebo model 16

# **: INTRODUCTION**

## Open Dynamic Robot Initiative

The project aims to build a low cost and low complexity actuator module using brushless motors that can be used to build different types of torque-controlled robots with mostly 3D printed and off-the-shelves components. Two works has been published under this project, one describing the actuator module and the quadruped design [1] and the other describing Tri-Finger Manipulator Platform and real-time reinforcement learning experiments [2]. The work is done in collaboration between the Motion Generation and Control Group, the Dynamic Locomotion Group and the Robotics Central Scientific Facility at the Max-Planck Institute for Intelligent System , the Machines in Motion Laboratory at New York University's Tandon School of Engineering and the Gepetto Team at the LAAS/CNRS. All the hardware and software has been open sourced under the BSD 3-clause license so that the robots can easily be reproduced by other research laboratories.

|  |
| --- |
|  |
| *Figure 1: Open Dynamic Robot Initiative* |

The dictionary says that the word “quadruped” means an animal which has four feet, especially an ungulate mammal which primarily includes opposite arm and leg lifts for the movements that is interlimb coordination. There are several ways in which a four-legged robot can be designed. Quadruped robots imitate animal walking gait which has many advantages like moving on uneven landscapes and extremely rough territories due to high nonlinear dynamic characteristics. To perform real-time operations, quadruped robots must have good stability which is a hot research topic in the field of robotics. To achieve the stability of the robot, most widely adopted path is the design and optimization of the structure of the robot or improve the control algorithm. A robot that moves on wheels has difficulties to overcome obstacles, but quadruped robots can adapt to avoid obstacles by adjusting its height. The quadruped robot has four leg with each leg having two degrees of freedom. quadrupeds can change their locomotive patterns, e.g. walking, trotting, and galloping to adopt the most energy-efficient gait for a given speed. Interlimb coordination mechanism works on autonomous decentralized control.

## Objective

The main objective in this work is to develop a ROS based simulator for Open Dynamic Robot Initiative. The Open robotics quadruped robot consists of eight identical actuator modules, four lower legs with foot contact sensors, IMU sensor and battery monitor. To navigate obstacles the quadruped will rely on simple depth data retrieved from a depth sensor (LiDAR/Camera). This will be limited to immediate collision detection, as this project will not go into depth about mapping. Positional data should be limited to registered depth data and inputted movement data. Orientation is cross referenced from IMU data force vector.

# **: BACKGROUND**



## Related Works

## ROS – Robot Operating System

ROS (Robot Operating System) is an open-source, meta-operating system commonly used for robotic development. It provides a vast collection of tools and packages allowing for rapid development of solutions without having to build systems from the ground up. This project employs the ROS Melodic branch, a ROS1 version 2 years old that is widely used and opted to avoid the newest branch Noetic Ninjemys as it is new, with less supported packages.

## ROS Master

The ROS Master started through roscore is the name server for communication and node-to-node connections. This can run locally (which is the default) or run remotely on a separate station, this utility of communication is one of the strong suits of ROS, as it allows a strong station to run heavy loads remotely, instead of running them locally on the robotic unit, saving processing power.

## Nodes, Subscribers and Publishers

A node in ROS can be likened to an executable program. They are typically charged with handling a single task, such as sensors, movement commands or control. ROS gives the advantage of having multiple small tasks solved in neatly arranged individual programs. Then passing the inputs, checks, or outputs between these different nodes trough the ROS messaging system. These messages are sent from a Publisher, being the origin point, or received at a Subscriber trough the topics. Do note that is possible to directly pass arguments to a topic from terminal, commonly used for testing. The topics themselves are channels in the ROS messaging framework that run in the background. These can be listened in to (for instance by using subscribers) or passed information if the topic has been initialized.

## Launch Files

Launch files is the premier way of running multiple ROS nodes with the necessary configuration easily. The launch files used in this project contains the required nodes, their arguments, file paths, initial configuration and so on. Without launch files arguments would have to be hardcoded into the nodes themselves or added into the terminal window at execution. They also prevent the need for an individual terminal window for each node, as the launch file can allow multiple nodes to be executed in the same terminal. Without this functionality it would not be strange to require dozens of terminals to execute a full package. It also prevents pathing issues caused by multiple terminals or poor sourcing of values.

## RViz

RViz is provided in ROS to visualize data selectively or as a whole. The 3D view shows models, TF’s, sensor data and more. Of note in the RViz window is the 2D Nav Goal tool commonly used to pass target navigation data to the system. With proper classification of data, it is also possible to use RViz for mapping of terrain and obstacles, though this relies on input or sensor data. A weakness of RViz is the limited physics representation. This project therefore uses Gazebo to give the values of the robot representative physics and their simulation.

|  |
| --- |
|  |
| *Figure 2: RViz Window Running Test Map* |

# **: 3-DIMENSIONAL MODELLING**



## Preparing the Mesh

Right-hand coordinate system is used in gazebo where +Z is up (vertical), +X is forward (into the screen), and +Y is to the left. The following steps were done to import the model to rviz and Gazebo.

* The first step is to center the mesh with center of mass-volume at (0,0,0) and orient the front (which can be subjective) along the x-axis.
* Add the desired material for the individual parts and export to COLLADA(.dae) format.
* Calculate the moment of inertia for the individual parts.

## Tools

A set of open source tools were used for generating the 3-dimensional models for the quadruped robot parts. For 3-D modelling and exporting models to COLLADA format Blender is used, which is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality and computer games. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, video editing and 2D animation pipeline. Physical inertial parameters such as the mass, center of mass location, and the moment of inertia matrix of all links are required for an accurate simulation. MeshLab is used for calculation of physical parameters and it is an open source application for processing and editing 3D triangular meshes. In addition to this, it provides a set of tools for editing, cleaning, healing, inspecting, rendering, texturing, and converting meshes.

## Quadruped Robot 3-Dimensional Models

|  |  |
| --- | --- |
|  | |
| *Figure 3: Body of quadruped robot* | |
|  |  |
| *Figure 4: Hips* | *Figure 5: Upper leg* |
|  |  |
| *Figure 6: Lower leg* | Figure : Foot |

The Quadruped robot consists of a main body and 4 identical legs consisting of a hip, an upper leg, a lower leg and a foot as shown in *Figure 2* to *Figure 5*. The models are centered to origin with center of mass(volume) as shown in the above figures. The units are scaled to metric units in Blender since the Gazebo uses the metric system. The information about materials and texture are added to the model using Blender. The mesh was exported as a COLLADA file when it was properly prepared. This format will contain both the 3D information and the materials whereas, the STL format contains only 3D information.



## Physical Parameters

The moments of inertia represent the spatial distribution of mass in a rigid body which is affected by the mass, size, and shape of a body. For applying the physical properties in Gazebo, the inertias and mass of the model is important. The moments of inertia can be expressed as the components of a symmetric positive-definite 3x3 matrix, with 3 diagonal elements (Ixx, Iyy and Izz), and 3 unique off-diagonal elements (Ixy, Ixz and Iyz). The moments of inertia are proportional to mass but vary in a non-linear manner with respect to size. Additionally, there are constraints on the relative values of the principal moments that typically make it much more difficult to estimate moments of inertia than mass or center of mass location. This difficulty motivates the use of software tools for computing moment of inertia.

In this work MeshLab was used to calculate the moments of inertia matrix. The below steps where followed:

* Import the mesh file in MeshLab: File -> Import
* To compute the inertial parameters, enable the display for Layers dialog which opens a panel in the right bottom of the application window: View->Show Layer Dialog.
* Command MeshLab to compute the inertial parameters by selecting: Filters->Quality Measure and Computations->Compute Geometric Measures.

## MeshLab reports

The reports from MeshLab for computing geometric measures are shown below. All the units are in kg/mm2. It has been observed that each inertia matrix is defined relative to a coordinate frame or set of axes. Diagonalizing the matrix yields its principal moments of inertia (the eigenvalues) and the orientation of its principal axes (the eigenvectors).

|  |
| --- |
| Body |
| Opened mesh /home/abraham/catkin\_ws/src/quadruped\_robot\_description/models/meshes/body\_rev04.dae in 69 msec  All files opened in 12113 msec  OUT OF SCOPE  Mesh Bounding Box Size 300.000000 600.000000 99.999977  Mesh Bounding Box Diag 678.232971  Mesh Bounding Box min -150.000000 -300.000000 0.000036  Mesh Bounding Box max 150.000000 300.000000 100.000015  Mesh Surface Area is 486777.812500  Mesh Total Len of 306 Edges is 19179.707031 Avg Len 62.678780  Mesh Total Len of 306 Edges is 19179.707031 Avg Len 62.678780 (including faux edges))  Thin shell (faces) barycenter: 0.000021 -0.000028 50.000038  Vertices barycenter 0.000122 -0.000116 50.000034  Mesh Volume is 16481011.000000  Center of Mass is 0.000025 -0.000008 50.000023  Inertia Tensor is:  | 505679347712.000000 -749.922974 -758.748779 |  | -749.922974 120063344640.000000 -23264.822266 |  | -758.748779 -23264.822266 601064800256.000000 |  Principal axes are:  | -0.000000 -1.000000 -0.000000 |  | -1.000000 0.000000 -0.000000 |  | -0.000000 -0.000000 1.000000 |  axis momenta are:  | 120063344640.000000 505679347712.000000 601064800256.000000 |  Applied filter Compute Geometric Measures in 81 msec |
| Hip |
| Opened mesh /home/abraham/catkin\_ws/src/quadruped\_robot\_description/models/meshes/shoulder\_rev01.dae in 50 msec  All files opened in 2846 msec  OUT OF SCOPE  Mesh Bounding Box Size 20.000004 100.000000 200.000015  Mesh Bounding Box Diag 224.499466  Mesh Bounding Box min -9.600773 -50.000000 -200.000000  Mesh Bounding Box max 10.399231 50.000000 0.000015  Mesh Surface Area is 50827.972656  Mesh Total Len of 501 Edges is 12694.997070 Avg Len 25.339315  Mesh Total Len of 501 Edges is 12694.997070 Avg Len 25.339315 (including faux edges))  Thin shell (faces) barycenter: 0.030701 0.000001 -99.811050  Vertices barycenter 2.647750 0.000002 -123.332741  Mesh Volume is 352503.531250  Center of Mass is 0.000004 -0.000000 -93.876007  Inertia Tensor is:  | 1370901120.000000 0.333347 -7898235.000000 |  | 0.333347 1090581376.000000 -1.537936 |  | -7898235.000000 -1.537936 303707648.000000 |  Principal axes are:  | 0.007400 0.000000 0.999973 |  | -0.000000 1.000000 -0.000000 |  | 0.999973 0.000000 -0.007400 |  axis momenta are:  | 303649216.000000 1090581376.000000 1370959616.000000 |  Applied filter Compute Geometric Measures in 71 msec |
| Upper leg |
| Opened mesh /home/abraham/catkin\_ws/src/quadruped\_robot\_description/models/meshes/thigh\_rev01.dae in 58 msec  All files opened in 2823 msec  OUT OF SCOPE  Mesh Bounding Box Size 119.999969 315.974304 57.000000  Mesh Bounding Box Diag 342.766327  Mesh Bounding Box min -81.503052 -178.070908 -57.000000  Mesh Bounding Box max 38.496922 137.903397 0.000000  Mesh Surface Area is 60429.312500  Mesh Total Len of 4644 Edges is 28364.146484 Avg Len 6.107697  Mesh Total Len of 4644 Edges is 28364.146484 Avg Len 6.107697 (including faux edges))  Thin shell (faces) barycenter: -0.810636 -2.742111 -28.499887  Vertices barycenter 17.446474 -111.987297 -28.543938  Mesh Volume is 468418.281250  Center of Mass is 0.000076 -0.000511 -28.499901  Inertia Tensor is:  | 4243598592.000000 839043008.000000 3623.758789 |  | 839043008.000000 525609024.000000 -6399.090332 |  | 3623.758789 -6399.090332 4577473536.000000 |  Principal axes are:  | 0.210401 -0.977615 -0.000002 |  | 0.977615 0.210401 -0.000014 |  | -0.000014 -0.000001 -1.000000 |  axis momenta are:  | 345031520.000000 4424176128.000000 4577473536.000000 |  Applied filter Compute Geometric Measures in 54 msec |
| Lower leg |
| Opened mesh /home/abraham/catkin\_ws/src/quadruped\_robot\_description/models/meshes/shank\_rev01.dae in 34 msec  All files opened in 2687 msec  OUT OF SCOPE  Mesh Bounding Box Size 47.500000 270.000000 39.949844  Mesh Bounding Box Diag 277.041962  Mesh Bounding Box min -23.549850 -148.322006 -39.974922  Mesh Bounding Box max 23.950150 121.678001 -0.025080  Mesh Surface Area is 30280.183594  Mesh Total Len of 1323 Edges is 16195.915039 Avg Len 12.241810  Mesh Total Len of 1323 Edges is 16195.915039 Avg Len 12.241810 (including faux edges))  Thin shell (faces) barycenter: -0.087347 -11.629372 -20.000055  Vertices barycenter -1.933211 -69.353226 -20.000000  Mesh Volume is 162465.906250  Center of Mass is 0.000001 -0.000065 -20.000004  Inertia Tensor is:  | 990160768.000000 24015334.000000 -0.307513 |  | 24015334.000000 23989654.000000 18.340973 |  | -0.307513 18.340973 990223104.000000 |  Principal axes are:  | 0.024833 -0.999692 0.000000 |  | -0.000000 0.000000 1.000000 |  | -0.999692 -0.024833 -0.000000 |  axis momenta are:  | 23393098.000000 990223104.000000 990757312.000000 |  Applied filter Compute Geometric Measures in 72 msec |
| Foot |
| Opened mesh /home/abraham/catkin\_ws/src/quadruped\_robot\_description/models/meshes/foot\_rev01.dae in 150 msec  All files opened in 4903 msec  OUT OF SCOPE  Mesh Bounding Box Size 19.000000 50.000000 50.000000  Mesh Bounding Box Diag 73.218849  Mesh Bounding Box min -9.500002 -25.000000 -50.000000  Mesh Bounding Box max 9.499999 25.000000 0.000000  Mesh Surface Area is 8262.914062  Mesh Total Len of 17472 Edges is 27799.205078 Avg Len 1.591072  Mesh Total Len of 17472 Edges is 27799.205078 Avg Len 1.591072 (including faux edges))  Thin shell (faces) barycenter: -0.000002 -0.000000 -24.999504  Vertices barycenter -0.000002 -0.000000 -25.000000  Mesh Volume is 34996.500000  Center of Mass is -0.000002 -0.000000 -25.000000  Inertia Tensor is:  | 10318125.000000 0.002293 -1.655262 |  | 0.002293 6166050.500000 -0.009906 |  | -1.655262 -0.009906 6166053.500000 |  Principal axes are:  | -0.000000 -0.999972 -0.007511 |  | -0.000000 0.007511 -0.999972 |  | 1.000000 0.000000 -0.000000 |  axis momenta are:  | 6166052.000000 6166053.500000 10318126.000000 |  Applied filter Compute Geometric Measures in 76 msec |

# **: ROS SIMULATION**



## Introduction

The Robot Operating System (ROS) can be defined as a collection of software frameworks and tools that can provide functionality of an operating system on a heterogeneous computer cluster but not an actual operating system. In this work, the tools which were used from ROS are listed below:

* catkin: The build system for ROS which is based on CMake. The dependencies used for creating the packages in the work includes rospy, rviz, controller\_manager, gazebo\_ros, joint\_state\_publisher and robot\_state\_publisher.
* roslaunch: Tool for launching multiple nodes in ROS as well as setting parameters on the ROS parameter server.
* rviz: Graphical interface tool or a 3D visualizer for visualizing information about simulated robots using plugins for many types of available topics.
* gazebo: Simulator for creating 3D worlds with physics and complex sensors.
* rqt – Main Graphical interface tool for ROS which hosts a wide variety of plugins. It is used in this work, for publishing commands through topic to the quadruped robot simulator in gazebo.



## Structure of ROS Package.

The package was created with necessary dependencies which consists of sub directories “launch”, “models”, “rviz\_config”, “config”, “urdf” and “worlds”. Inside the “worlds” directory, a subdirectory named “meshes” were created followed by copying the created 3D models of quadruped robot. Inside the “urdf” directory file named “quadruped\_robot\_geometric\_rev02.urdf” serves as the backbone of the model defining the robot name, links, joints, and transmission. The “launch” directory consists of file named “urdf\_visualize.launch” to load the quadruped robot to rviz. It also consists of following files named “spawn\_urdf.launch”, “quadruped\_robot\_control.launch”, “spawn\_quadruped\_robot\_control\_with\_controllers.launch” to spawn the quadruped robot model in urdf to gazebo which are called using the main launch file “start\_quadruped\_robot\_control\_with\_controllers.launch”. The PI control parameters are defined in file “quadruped\_robot\_controller\_rev00.yaml” under “config” directory.

## Model Anatomy

The quadruped robot model consists of a main body link which is attached to four hip links located at four corners using a fixed joint. The individual hip links are connected to upper leg link using revolute joint with limit from -1.5 to +1.5 radians. A revolute joint with limit from -3.0 to +3.0 radians joints the upper leg link and the lower leg link. Finally, the foot link is attached with the lower leg link using a fixed joint. A total of eight motors is interfaced to the eight individual revolute joints.

## Simulation of Model in rviz

The following steps were followed to visualize the fully assembled model of quadruped robot in rviz.

* Open the workspace in terminal using command: cd catkin\_ws
* Setup the environment variables: source devel/setup.bash; export LC\_NUMERIC="en\_US.UTF-8";
* Open the src directory and launch “urdf\_visualize.launch”: cd src; roslaunch quadruped\_robot\_description urdf\_visualize.launch model:='$(find quadruped\_robot\_description)/urdf/quadruped\_robot\_geometric\_rev00.urdf'
* Load the rviz config file “test.rviz” from “rviz\_config” dircetory.

The results from rviz simulation are shown in *Figure 7*.

|  |
| --- |
|  |
| *Figure 8: rviz sinulation results for quadruped robot* |

## Simulation of Model in Gazebo

Three terminals are required to run the simulation in gazebo. One to load the gazebo empty world with gravity, another one to launch the quadruped robot to the gazebo and third one to launch the GUI - rqt\_gui for publishing commands to topics. The following steps were followed to visualize the fully assembled model of quadruped robot in Gazebo.

* In terminal 1:
  + Open the home directory: cd
  + Set the path: ROS\_PACKAGE\_PATH=$HOME/catkin\_ws:/opt/ros/noetic/share
  + Update the latest quadruped\_robot\_description files in gazebo directory: rm -rf /home/abraham/.gazebo/models/quadruped\_robot\_description; cp -r /home/abraham/catkin\_ws/src/quadruped\_robot\_description /home/abraham/.gazebo/models/
  + Launch the empty world with gravity in gazebo: roslaunch gazebo\_ros empty\_world.launch

|  |
| --- |
|  |
| *Figure 9: rqt\_gui window* |

* In Terminal 2:
  + Open the workspace directory: cd catkin\_ws
  + Setup the environment variables: source devel/setup.bash
  + Open the “src” directory: cd src
  + Delete any existing robot with same name: rosservice call /gazebo/delete\_model "model\_name: 'quadruped\_robot\_project'"
  + Launch the quadruped robot to gazebo: roslaunch quadruped\_robot\_description start\_quadruped\_robot\_control\_with\_controllers.launch
* In Terminal 3:
  + Launch the GUI and add the commands as shown in *Figure 8*: rosrun rqt\_gui rqt\_gui

The results from simulation in gazebo are shown in *Figure 9*.

|  |
| --- |
|  |
| *Figure 10: Gazebo simulation results for quadruped robot* |

## Challenges in Transition of Quadruped Robot to Champ

Even though physical transition of Quadruped Robot to Champ has been successfully emulated in this work, but mass and inertia needs further tuning. Actuators and transmission did not meet the expected behavior with the Champ framework. It was found that Champ framework relies on an additional joint that was not in quadruped robot used inopen dynamic robot initiative, which causes some issues with translation and requires further overhaul of the champ framework.

|  |
| --- |
|  |
| *Figure 11: Gazebo running in test world with obstacles* |
|  |
| *Figure 12: RQT Graph showing nodes and topic connections* |

Camera is added and enabled, but the changes to the urdf model had some negative impact on the RViz presentation and both LiDAR and camera needs further tuning of parameters.

|  |
| --- |
|  |
| *Figure 13:Camera feed from gazebo model* |

# **:** **FUTURE WORK**



## Improvement in model

## Controller Tuning

Tuning of PID parameters for the given physical model is important and can be done in future work for actuators and transmissions to show up the expected behavior.

## Implementation of Algorithms for GAIT

The current gait controls used in the Champ robot (Ref?) simply applies a set of variables to control speed and limit angles of movement. It does so by using the following:

* Knee orientation - How the knees should be bent. Configure for following orientation .>> .>< .<< .<> where dot is the front side of the robot.
* Max Linear Velocity X (meters/second) - Robot's maximum forward/reverse speed.
* Max Linear Velocity Y (meters/second) - Robot's maximum speed when moving sideways.
* Max Angular Velocity Z (radians/second) - Robot's maximum rotational speed.
* Stance Duration (seconds) - How long should each leg spend on the ground while walking. The higher the stance duration the further the displacement is from the reference point.
* Leg Swing Height (meters) - Trajectory height during swing phase.
* Leg Stance Height (meters) - Trajectory depth during stance phase.
* Robot Walking Height (meters) - Distance from hip to the ground while walking.
* CoM X Translation (meters) - Move the reference point in the X axis. This is useful when you want to compensate for the weight if the center of mass is not in the middle of the robot (from front hip to rear hip).
* Odometry Scaler - A multiplier to the calculated velocities for dead reckoning. Normally this value ranges from 1.0 to 1.20.

This is limiting for developing a solid gait or applying different sets of gaits. To further increase the possible sets of movements this system would require an overhaul. Another issue is the adaptability issue, as the robot is limited to a set of values, instead of having a feedback to correct these values. Using learning algorithms and neural networks to develop better default gaits, and further to change gait on the move would drastically increase the range of movement of the robot. For developing a standard gait this approach would not cause many issues, but for on the fly change a new limiting factor is on-time computation. Instead looking into a few sets that are optimized for several scenarios and knowing when to change between them would be preferential. However, if this is part of a neural network these sets can be selected as needed.

# **APPENDIX I: REFERENCES**

# **APPENDIX II: SOURCE CODE**