Development of a neuro-inspired control system for quadrupeds to emulate sensorimotor processes in animals

A project report

submitted by

SHREYAS SHANDILYA

in partial fulfilment of requirements

for the award of the dual degree of

BACHELOR OF TECHNOLOGY IN

ENGINEERING DESIGN

AND

MASTER OF TECHNOLOGY IN

BIOMEDICAL DESIGN

DEPARTMENT OF ENGINEERING DESIGN

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MARCH 2021

CERTIFICATE

This is to certify that the project titled **Development of a neuro-inspired control system for quadrupeds to emulate sensorimotor processes in animals**, submitted by **Mr Shreyas Shandilya**, to the Indian Institute of Technology Madras, for the award of the degrees of **Bachelor of Technology in Engineering Design** and **Master of Technology in Biomedical Design**, is a *bona fide* record of the research work done by her under my supervision. The contents of this project, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr Asokan Thondiyath
Project Adviser
Professor
Department of Engineering Design
Indian Institute of Technology Madras
Chennai 600 036

Dr V.S. Chakravarthy
Project Adviser
Professor
Dept. of Biotechnology
IIT Madras, 600 036

ACKNOWLEDGEMENTS

ABSTRACT

KEYWORDS: Computational Neuroscience, Neural Networks, Central Pattern Generators, Deep Learning, Robotics, ROS

This project report document studies towards the development of a neural network control system for quadruped robots. Locomotion and navigation require the perception of the surrounding space and embodied decision making to make an appropriate response to an incoming stimulus. Animals are highly adept at such tasks, and the extraordinary agility and dexterity exhibited by animals are highly desirable for legged robots. A hybrid neural network for the locomotion and navigation control system of a quadruped robot is proposed. The proposed system leverages oscillatory models of brain function to mimic a multi-level multi-loop hierarchy of locomotion control. The lower level is controlled by Central Pattern Generators in the spinal cord and the higher levels by the brain. This project focuses on emulating gait with navigation when the hierarchical controller is trained using Reinforcement Learning and the developed control system's consequent deployment on a quadruped.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

FK	Forward Kinematics
IK	Inverse Kinematics
DL	Deep Learning
RL	Reinforcement Learning
DoF	Degree-of-freedom
CPG	Central Pattern Generator
DNN	Deep Neural Network
DIY	Do It Yourself
IMU	Inertial Measurement Unit
URDF	Universal Robot Description Format
Osc.	Oscillators
MSE	Mean Squared Error
ZMP	Zero Moment Point
CoM	Center of Mass
RNN	Recurrent Neural Network

NOTATIONS

$\vec{r}_i$	The *i*th link vector of the MePed knee four bar linkage
θ_i	The *i*th joint angle position of the MePed knee four bar linkage
ω_i	The *i*th joint angular velocity point of the MePed knee four bar linkage
α_i	The *i*th joint angular acceleration point of the MePed knee four bar linkage
v_i	Velocity of the *i*th link of the MePed knee four bar linkage
a_i	Acceleration of the *i*th link of the MePed knee four bar linkage
T	Maximum torque from TowerPro SG90 Micro Servo
F_{ijx}	Force on the *j*th link from *i*th link along the x axis
F_{ijy}	Force on the *j*th link from *i*th link along the y axis
F_{ix}	Net Force on the *i*th link along the x axis
F_{ix}	Net Force on the *i*th link along the y axis
C_i	Net Torque acting on *i*th link
L_T	Length of the MePed Thigh
L_i	Length of *i*th link of the MePed knee four bar linkage
ϕ	Angle between $\vec{r}_0$ and the x-axis
$\vec{a}$	Resultant acceleration of the quadruped during gait
$\vec{\alpha}_R$	Resultant angular acceleration of the quadruped during gait
m	Mass of the Meped
I_R	Moment of Inertia of the MePed about the center of mass position at rest
$\vec{N}$	Normal Reaction forces acting on the quadruped legs
$\vec{F}_{fr}$	Friction forces acting on the quadruped legs
z	Complex valued dependent variable of the Hopf Oscillator
x	Real part of the dependent variable of the Hopf Oscillator
y	Imaginary part of the dependent variable of the Hopf Oscillator
ω	Frequency of a Hopf Oscillator
μ	Amplitude controlling parameter of a Hopf Oscillator
T_{sw}	Swing period of a gait cycle
T_{st}	Stance period of a gait cycle
β	Duty Factor of a gait cycle
θ_h	Amplitude of oscillation of Hip Joint Trajectory
θ_k	Amplitude of oscillation of Knee Joint Trajectory

CHAPTER 1

INTRODUCTION

1.1 Background

Animals are highly adept at locomotion and navigation under challenging terrains, capable of responding to a sudden stimulus with extraordinary agility and dexterity. They exhibit behaviour like gait switching, rapid acceleration and deceleration, evasive manoeuvres, climbing, jumping and search. The seamless transition between different behaviours and agile response to an incoming stimulus resulted from the evolution of neural pathways for adaptive locomotion, perception and embodied decision making. A model of such integrated sensorimotor processing can provide greater autonomy and deftness to robotic locomotion and navigation.

Among animals, quadrupedal locomotion is the most common, with some animals capable of limited bipedal locomotion. Such prevalence of quadrupedal locomotion in nature may be attributed to the inherent stability and wide range of available configurations of locomotion. These characteristics of quadrupedal locomotion make it suitable for several applications such as last-mile delivery and search and rescue to the potential to work in unstructured, severe and dangerous environments. Consequently, several quadruped robots have been developed over the years. The TITAN series [9], Tekken IV [10], MIT Cheetah [11] and BigDog [12] have been some of the successful quadruped robots capable of gait in severe and difficult environments. The aforementioned quadruped robots' critical technologies attributed to their success are their biomimetic structure, a high power density of actuator, real-time control methods, and integrated environment perception [13]. All the developments through these quadrupeds lead to an integrated environment perception and decision-making system.

Neuroscientific models of Cognition have explained information processing in living organisms at varying degrees of details. Available models range from bio-molecular models operating at the synaptic levels to aggregate oscillatory models operating at the level of neuron ensembles. Since animal behaviour is desired in quadrupeds, neuroscientific models of sensorimotor processes to emulate such behaviour seem logical.

Moreover, such models also benefit from the massively parallel and efficient computations seen in the brain, making such models ideal for robotics' resource-constrained applications.

This report documents the design and development of a neural network architecture for autonomous control. The neural network combines Central Pattern Generator(CPG) theory with Deep Neural Networks to emulate different aspects of a hierarchical multi-loop sensorimotor processing with a one-to-one mapping between the proposed architecture modules and the nervous system's involved processing centres. Furthermore, a Deep Deterministic Policy Gradient is used to learn sophisticated behaviour with experience. An incremental methodology is followed to learn increasingly complex behaviour by the formulation of appropriate reward functions. The developed neural network is tested and deployed on a prototype quadruped platform, built as a part of this project.

1.2 Motivation

The primary motivation for this work is the need for adaptive gait generation for an efficient control strategy for legged robots based on biological locomotion principles. Animal locomotion requires multi-dimensional coordinated rhythmic patterns that need to be correctly modulated to satisfy multiple constraints such as generating forward motion with low energy, without falling over, adapting to possibly challenging terrain, and allowing the modulation of speed and direction [14]. In vertebrates, CPGs are the essential building blocks that generate and modulate the rhythmic patterns required for locomotion. The neuroscientific theory defines CPGs as non-linear dynamical systems with limit cycle behaviour given the dynamical system's parameters. CPGs have been successfully used to control a variety of robots. A distributed control strategy was proposed in [15] for a modular quadruped robot. The strategy proposed used N coupled amplitude-controlled Hopf oscillators to control the N degrees of freedom in the quadruped. Similarly, a four oscillator CPG model that integrates sensory feedback into the CPG for gait modulation was proposed in [3]. This model was further extended in [16] by introducing a two-level hierarchy, such that the higher level modulates the CPG model by appropriately modifying CPG parameters.

In biology, CPGs are activated and modulated by simple tonic signals from higher parts of the brain, and they are strongly coupled with the body they control and the environment via sensory feedback. The strong coupling via sensory feedback allows for integrated environment perception and embodied decision making. Although current CPG models are successful in gait generation, gait modulation and learning with CPGs remain challenging. Moreover, the integration of higher-level control with a CPG requires modulating high dimensional parameter vectors that determine CPG behaviour instead of simple tonic signals. Furthermore, there have been very few instances where a CPG is a part of the learning system. For instance, a CPG model was implemented as a layer in a deep neural network to facilitate backpropagation in [17]. However, in most cases, behaviour like gait transition and modulation is achieved using predefined relationships to produce appropriate CPG parameters. Due to such shortcomings in CPG models, learning with experience is challenging, and genuine animal-like behaviour can not be achieved.

On the other hand, Deep Neural Networks (DNNs) are capable of universal approximation and have standard learning rules for updating parameters. Gradient-based or gradient-free techniques may be used to optimise the parameters of a DNN. The optimisation technique used constrain the structure of DNN to a specific class of networks. For instance, backpropagation requires the network to have no loops, whereas CPG coupling almost always forms loops. Although the integration of DNNs with CPGs constrains network structure choice, the ability of universal approximation added to the network allows for the transformation of a low dimensional desired motion or state vector into a tonic signal that can be used to modulate the CPG for rhythm generation. The hierarchical controller proposed in [5] uses a DNN whose outputs are used to regulate selected CPG network parameters. Though the CPG is not a part of the learned network, the use of the DNN allows for the integration of a low-dimensional feedback signal for control.

Emulation of more sophisticated behaviour like obstacle avoidance and search can be achieved using Reinforcement Learning (RL) to learn an appropriate policy. For instance, a hierarchical controller was proposed in [18] that applies RL to negotiating obstacles with a quadruped robot. Since a DNN-CPG architecture requires only low dimensional desired motion or state vector to modulate the network output, a hierarchical controller similar to those proposed in [18] and [5] can be developed using RL to emulate a particular animal behaviour. Such a controller, combining the universal approximation and learning abilities of DNNs with the adaptive rhythm generation of CPGs, is expected to be successful in the nimble, autonomous control of a quadruped in challenging terrains.

1.3 Objectives

The primary aim of this work is to develop a generalised DNN-CPG architecture for autonomous control problems. The developed architecture is a model of the sensorimotor processing typical across vertebrates, and its functioning is to be demonstrated on a quadruped platform, also assembled as a part of this project. It is also intended to study the evolution of quadruped behaviour as it interacts with its environment. The developed model is limited to the emulation of the behaviour of obstacle avoidance and path planning.

1.4 Scope

The following have been achieved in the present work as direct objectives of its initiation.

1. Assembly of a Quadruped Platform for the testing and deployment of developed neural network
2. Development of ROS-Gazebo simulation environment for RL
3. Development of a DDPG for RL
4. Formulation reward functions to quantise different aspects of quadruped behaviour
5. Development of a DNN-CPG architecture for autonomous control

Additionally, the following are expected to result from the investigation of related topics.

1. An improved understanding of the nervous system and distributed processing seen in the brain
2. An improved understanding of quadruped kinematics and dynamics

1.5 Organisation of the Report

The remainder of this report is organised as follows. Chapter 2 describes the MePed Quadruped Platform, onto which the developed DNN-CPG is to be deployed. Chapter 3 describes the development of the DNN-CPG and the use of supervised learning to prove the utility of DNN-CPG for control applications. Chapter 4 describes the reward function formulation for RL. Chapter 5 describes the RL algorithm, the DDPG and how the DNN-CPG is modified for RL.

CHAPTER 2

THE MEPED QUADRUPED PLATFORM

2.1 Introduction

The MePed is a small open-source quadruped robot, $16cm \times 16cm \times 6cm$ in size with 8 DoF, two on each leg. Figure 2.1 and Figure 2.2 depict the MePed without the controller. It is compatible with both Arduino and Raspberry Pi and serves as the platform for testing and deploying the DNN-CPG models to be developed. The quadruped is currently controlled using an Arduino Mega 2560, with a V2 sensor shield. Tower Pro SG90 micro servo motors are used to actuate the joints. The robot chassis weighs 196.36 g and the complete quadruped, with servos and controller weighs 345.32g.
The Tower Pro SG90 servo produces a maximum torque of 1.5 $\mathrm{kg\,cm^{-1}}$ and an operating speed of 0.143 $\mathrm{s\,rad^{-1}}$ at $6V$ power supply. The crawling type quadruped, though with a lower ground clearance has greater stability, making it ideal for testing and deployment of the DNN-CPG architecture.

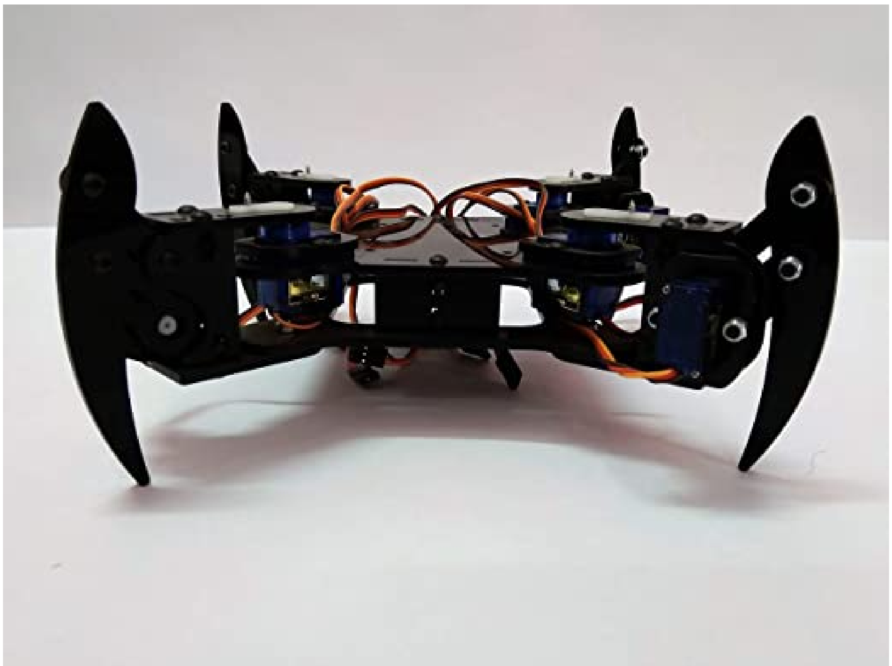


Figure 2.1: MePed Front View

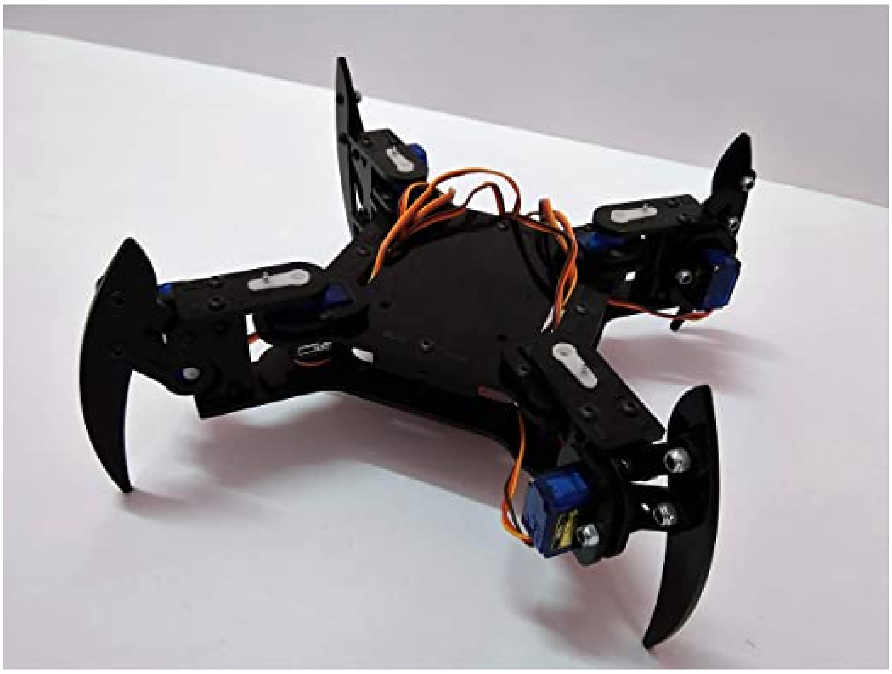


Figure 2.2: MePed Perspective View

The MePed comes as a DIY kit with the chassis and required screws. The Arduino Mega, the sensor shields and the servos were acquired based on the project's requirements. The Arduino Mega has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, four UARTS (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the sensor, communication and power supply modules needed for the quadruped. Moreover, it has 256 kB flash memory, 8 kB SRAM and 4 kB EEPROM which is just sufficient for a small neural network. The sensor shield provides enough pins to interface with all the peripherals to be mounted onto the quadruped.

The Sensor Shield provides unique pins for Ultrasonic Sensor Interface, Bluetooth Interface, SD card interface and a Wireless Module Interface. Figure 2.3 depicts the schematics of the sensor shield used. Of the available interfaces, Ultrasonic Sensor Interface and the Wireless Module Interface are currently occupied. The application of the ultrasonic sensor is described in detail in Section 2.2.1. The wireless module serves to interface the quadruped with a remote control or a computer for logging sensor information. A CMOS camera also interfaces with the sensor shield. Though there is no interface for the camera on the sensor shield, the camera connects to the controller by connecting to appropriate analog, PWM, power and ground pins.

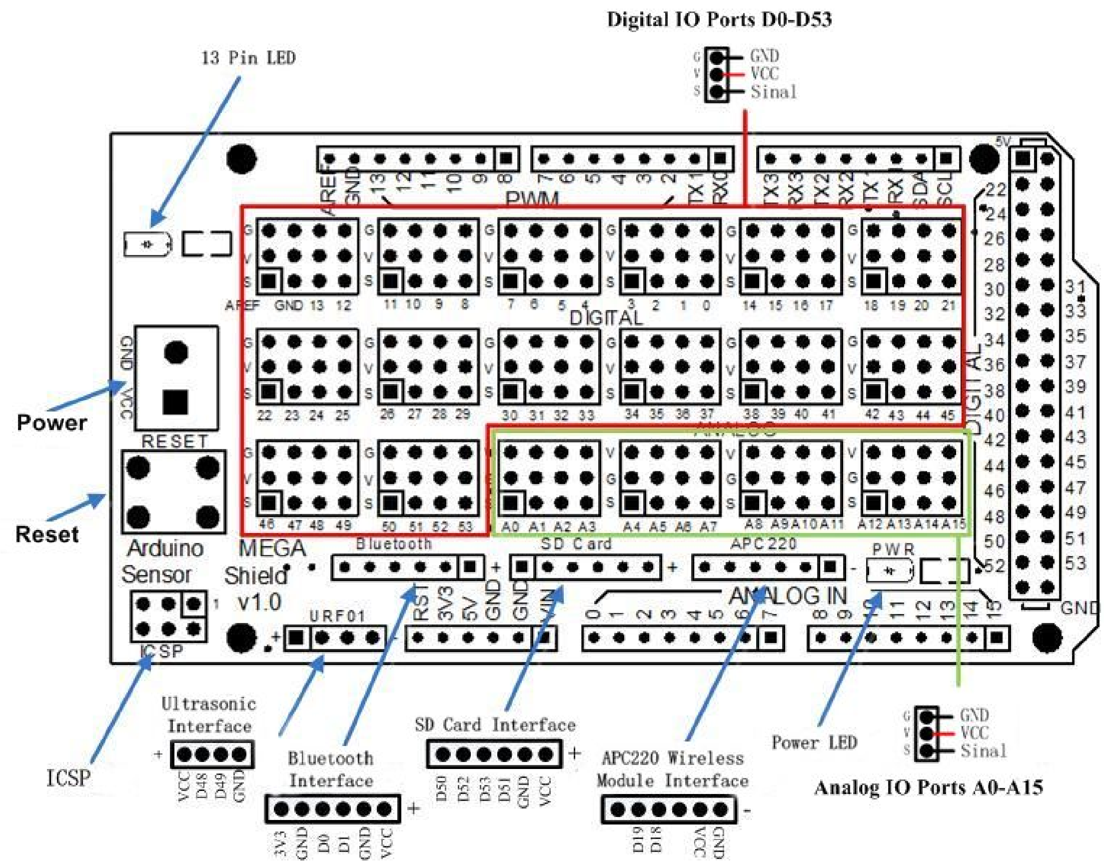


Figure 2.3: Sensor Shield V2 for Arduino Mega 2560

2.2 Sensors

The quadrupes is to be mounted with the following four sensors.

1. Ultrasonic Sensor
2. CMOS Camera
3. IMU Sensor
4. Contact Sensor

2.2.1 Ultrasonic Sensor

The ultrasonic sensor is used to detect the distance of the nearest object in front of the quadruped. The sensor emits short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo.

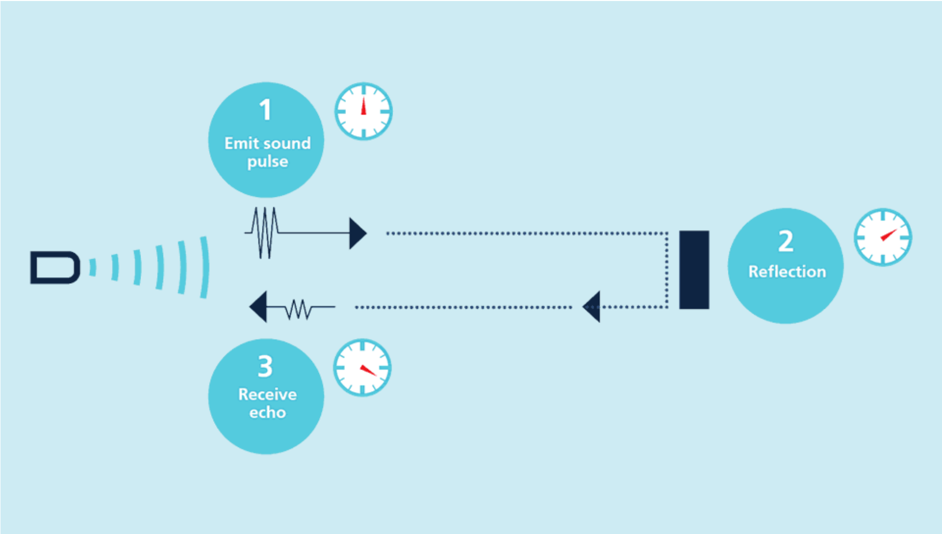


Figure 2.4: Working Principle Ultrasonic Sensor [1]

Figure 2.4 illustrates the working principle of the ultrasonic sensor. The sensor used can detect objects within a range of 2 cm to 450 cm and within an arc of $15°$.

2.2.2 CMOS camera

The REES52 OV7670 Robodo OV7670 640x480 VGA CMOS Camera Image Sensor is a high sensitivity color image sensor. Figure 2.5 depict the REES52 OV7670 Robodo CMOS Image Camera Sensor used. The sensor has high sensitivity for low-light operation, low operating voltage for embedded applications. The sensor operates at 15 fps and produces 640×480 images. The sensor also supports additional features such as image scaling, lens shading correction, flicker 50/60Hz auto detection and color saturation level auto adjust. The sensor provides vision capabilities to the DNN-CPG sensorimotor processing model.

Figure 2.5: REES52 OV7670 Robodo CMOS Image Camera Sensor

2.2.3 IMU

IMU is a 9-axis sensor that measures orientation, velocity, and gravitational forces by combining Accelerometer, Gyroscope, and Magnetometer into one. IMU is used to obtain the following values for the construction of a quadruped state.

1. Orientation
2. Linear Acceleration
3. Angular Velocity

An IMU is not currently mounted on the MePed, but is present in the simulation model and plays a crucial role in controller learning.

2.2.4 Contact Sensor

The contact or the bump sensor is used to detect collision of the legs with the ground. It serves to sense the stance and the swing legs of the MePed. The sensor works with the help of a switch which gives the system a touch effect to be used as a touch sensor. Though the bump sensor is not currently mounted on the MePed, it is present in the simulation model and plays a crucial role in controller learning.

2.3 Physics Simulation

2.3.1 CAD Model

Being a low-cost open-source hardware model, only the STL(Standard Tessellation Language) model of components were available for the MePed. Due to the need to formulate quadruped kinematics and dynamics, a detailed description of the quadruped physical characteristics, such as the mass, dimensions and centre of mass of the constituent parts, and the quadruped were needed. Moreover, such information was valuable for the development of a more accurate URDF description of the MePed. Figure

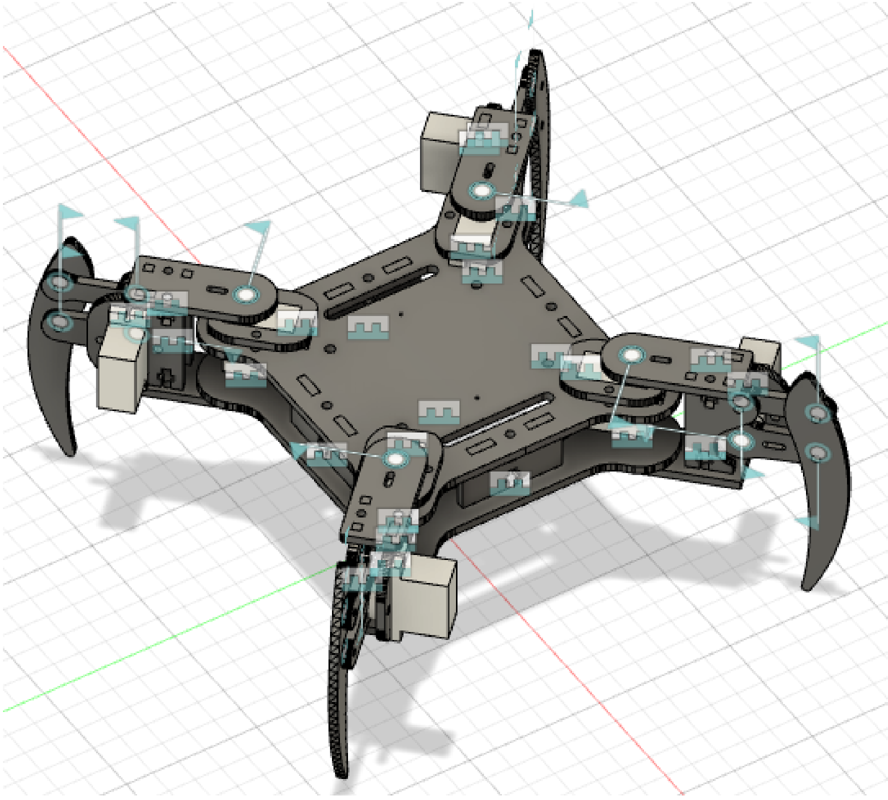


Figure 2.6: CAD model of the MePed assembled using AutoDesk Fusion 360

2.6 depicts the CAD model the MePed.

The MePed is 3D printed using a variant of Acrylic of density 0.00157 $\mathrm{g\,mm^{-3}}$. The CAD model of the MePed is available publicly[1].

Table 2.1: List of Components of the MePed Assembly and their physical attributes.

Component	Mass (g)	Bounding Box Dimensions ($L \times W \times H$ mm)
Body Bottom Plate	35.2	122.0×122.0×3.2
Body Spacer	4.2	38.3×3.2×25.5
Body Top Plate	51.8	122.0×122.0×3.2
Leg Rev	5.4	17.1×17.1×96.2
Leg Parallel Plate	3.1	19.3×19.3×42.8
Leg Parallel Linkage	1.6	28.2×28.2×11.3
Leg Servo Arm	2.1	29.9×29.9×15.9
Leg Servo Mount	4.03	24.8×24.8×42.8
Leg Servo Retainer	2.5	35.0×35.0×3.2
Leg Top Pivot Plate	4.4	43.8×43.8×3.2
Sensor Shield V2	40.0	100.0×54.0×18
Arduino Mega 2560	37.0	101.5×53.3×20
Tower Pro SG90 Micro Servo	9.0	32.3×32.3×30.0

[1]The CAD model can be found at the following link: https://a360.co/3eyVJS8

2.3.2 URDF Description

Universal Robot Description Format (URDF) is an XML based domain specific modelling language used to describe a robot's layout and appearance and to specify additional information for kinematic and dynamic description of the robot like joint limits, mass, friction values and so on. A URDF description is required for building any robotic simulation.

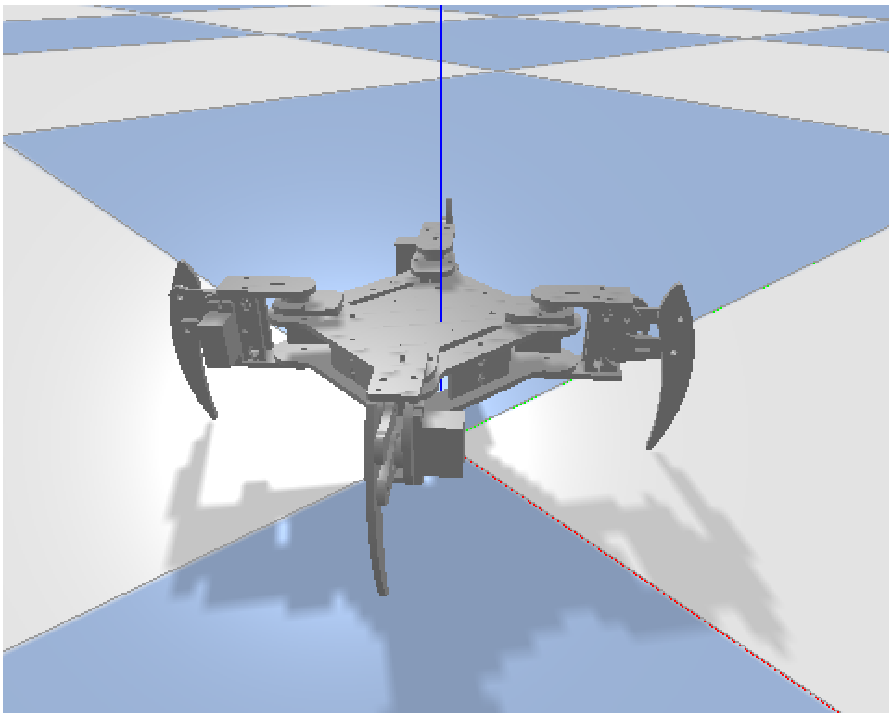


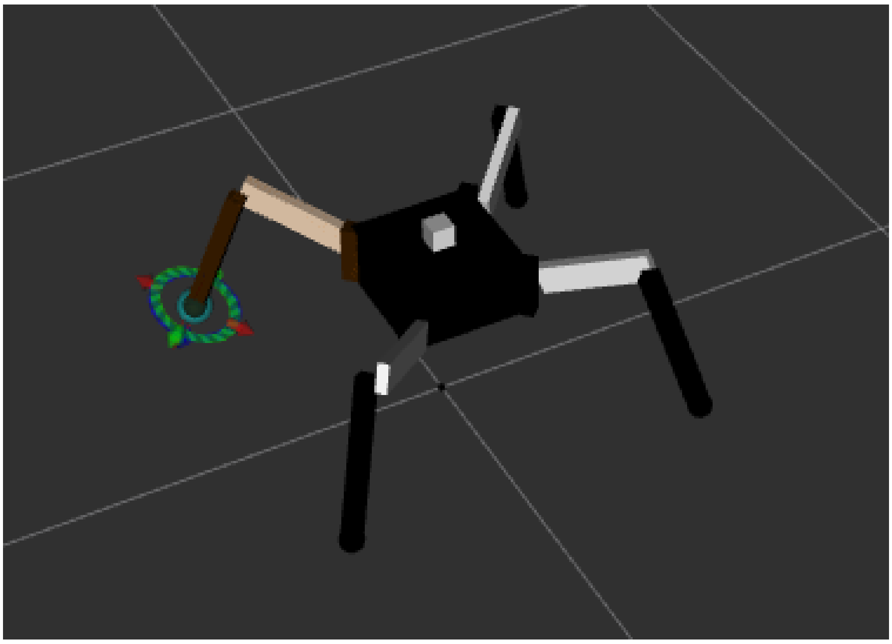
Figure 2.7: PyBullet simulation the MePed using URDF

The URDF description of the MePed was generated using a Fusion360 plugin called *fusion2urdf*[2]. Figure 2.7 depicts the URDF description of the MePed in PyBullet. The collision model was further modified to reduce the computational expense and time of computing the physics of the model. The modifications made involved the replacement of STL mesh files for generating collision boundaries with simple cuboidal shapes defined using the component bounding box dimensions in Table 2.1. Though a reduction in computational expenses and time was seen, further reduction were required for parallel execution of the physics engine and the DDPG for RL.

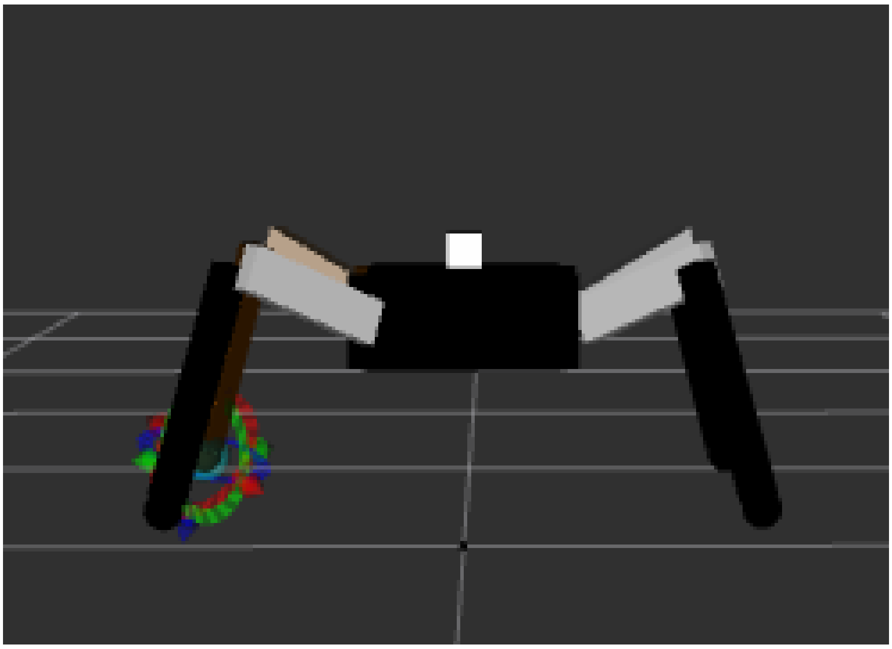
[2]The plugin is available at the following link: https://github.com/syuntoku14/fusion2urdf

2.3.3 Simplified Quadruped Description

Due to high computational expenses of the collision model of the MePed, it can not be directly used for training the DDPG. The high computational requirements are mainly due to the four bar linkages at the knee joints and a large number of small components interacting simultaneously. Thus, a simplified URDF description was developed for the MePed. Figure 2.8 illustrates the simplified URDF description.



(a) Perspective View



(b) Front View

Figure 2.8: Simplified URDF description of the MePed

The simplified description has 12 DoF instead of 8, as in the MePed. The additional degrees of freedom were introduced to emulate the kinematics of the MePed. Due to four-bar linkage as the knee joint, the end-effector motion for the MePed is not like a typical 8 DoF quadruped. Figure 2.9 illustrates the trajectories of the leg endpoints. For a typical 8 DoF quadruped, a knee joint movement of 30° would imply proportionate movement along the x, y and z axes. However, due to the four-bar linkage, the movement along the x and y axes is diminished. An additional joint was introduced in each leg to mimic the same behaviour in the simplified URDF description. The additional joint is constrained such that a trajectory similar to Figure 2.9 is obtained. For simplicity, the additional joint's movement is constrained to be 0.01 times the knee joint in the opposite direction. An unintended advantage of such a URDF description is the increased generalisability of the simulation environment developed.

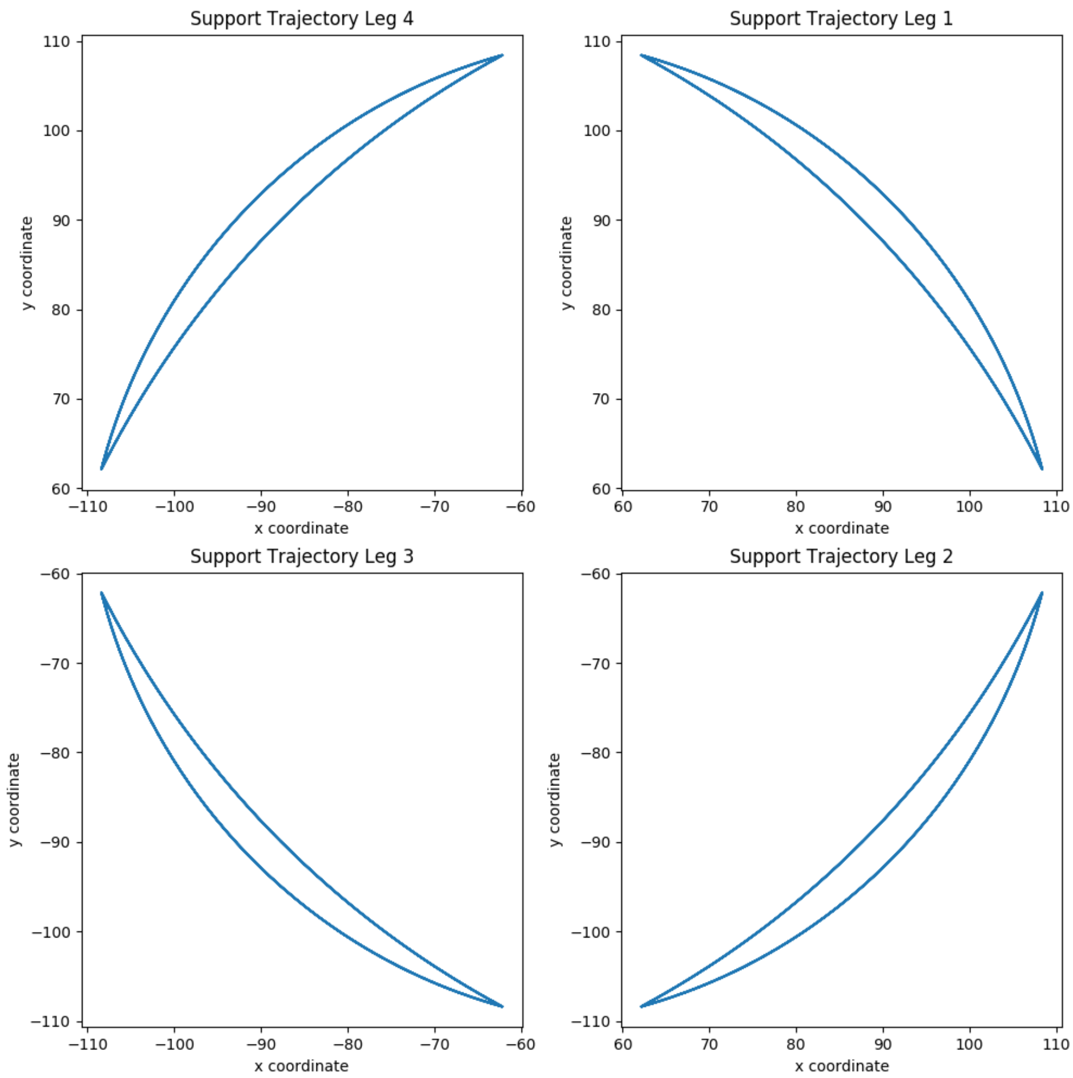


Figure 2.9: Leg End Point Trajectory in the XY plane for the MePed

2.4 Kinematics & Dynamics

Embodied Cognition spans the brain, body and environment. Perception and decision making involves the brain and the environment. A model of the body's physical behaviour must be known to include the body in the process of embodied decision making. The kinematics and dynamics of the MePed were studied to formulate the kinematics and dynamics of the simplified quadruped description.

2.4.1 MePed Kinematics & Dynamics

Quadruped motion results from forward propulsion caused by the friction forces as a leg in moves backwards during the stance phase of its gait cycle. Figure 2.10 depicts the forces acting on the quadruped due to the movement of its legs and the resulting acceleration. The resulting motion of the quadruped is not in a straight line, rather a lateral displacement also occurs as seen in Figure 2.11.

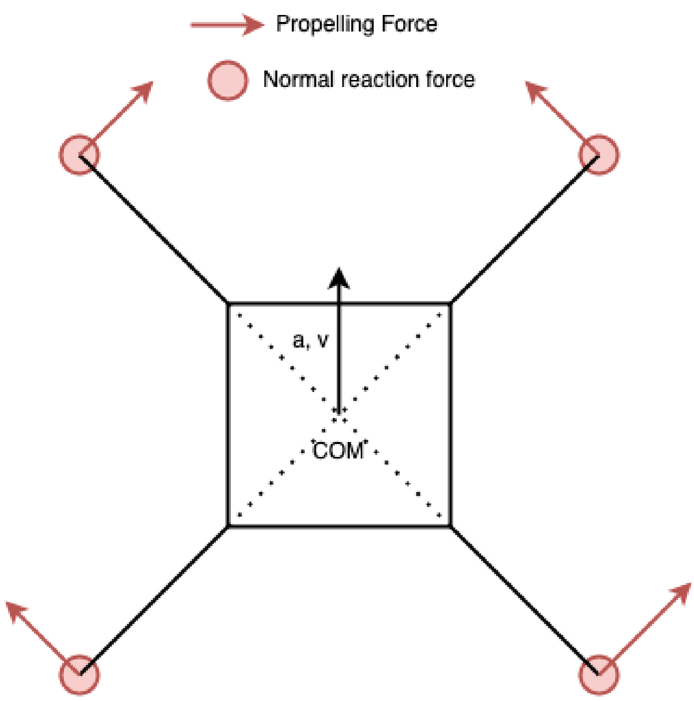


Figure 2.10: Simplified Model of MePed Dynamics as seen from the top

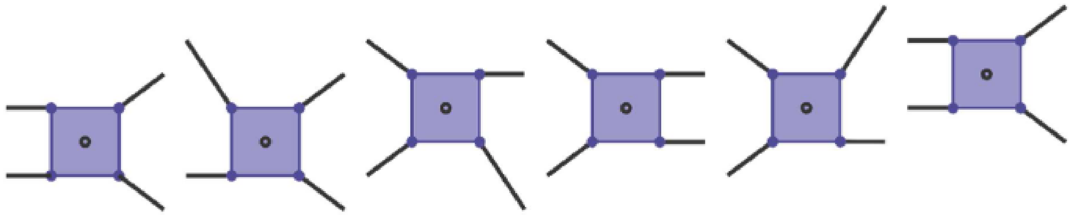


Figure 2.11: Quadruped Pose during a single period of Gait Cycle [2].

The kinematics and dynamics of the four-bar linkage in the knee were studied closely to determine the corresponding behaviour of the additional joint introduced in the simplified quadruped description. The four-bar linkage was fixed at the points F and O to the thigh. A TowerPro SG90 Micro Servo actuated the joint at O. Due to the knee joint's unique construction, a movement of an angle θ at O does not imply the same movement at the leg tip. Figure 2.12 illustrates the leg and all external and internal forces on the leg and the four-bar linkage at the knee. The equations 2.1 through 2.28 depicts the four-bar linkage's kinematics used to determine the behaviour of the additional joint in the simplified quadruped description.

$$l = \sqrt{|\vec{r_0}|^2 + |\vec{r_1}|^2 - 2|r_0||r_1|cos(\theta_1 - \phi + \pi)} \tag{2.1}$$

$$\beta_1 = \arcsin(\frac{|\vec{r_1}|}{l} sin(\theta_1 - \phi + \pi)) \tag{2.2}$$

$$\beta_2 = \arccos \frac{|\vec{r_2}|^2 + l^2 - |\vec{r_3}|^2}{2|\vec{r_3}|l} \tag{2.3}$$

$$\delta = \arcsin(\frac{l}{|\vec{r_3}|} sin\beta_2) \tag{2.4}$$

$$\theta_2 = \beta_2 - \beta_1 + \phi - \pi \tag{2.5}$$

$$\theta_3 = -\beta_1 - \delta + \phi - \pi \tag{2.6}$$

$$\vec{r_i} = [L_i \cos\theta_i, L_i \sin\theta_i, 0]^\top \quad \forall \quad i \in \{1, 2, 3\} \tag{2.7}$$

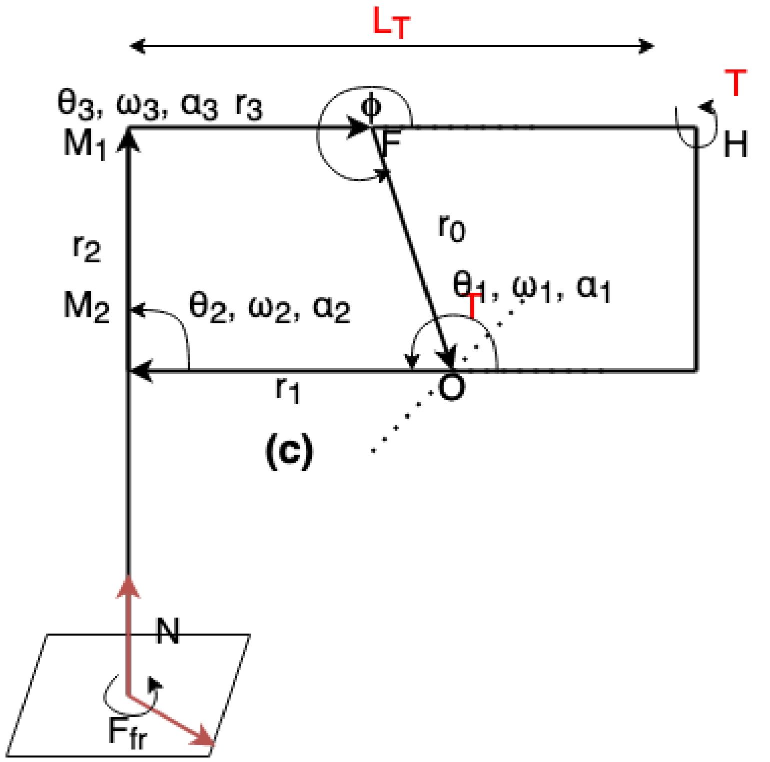
$$\vec{r_0} = [L_0 \cos\phi, L_0 \sin\phi, 0]^\top \tag{2.8}$$

$$\vec{r_1} + \vec{r_2} + \vec{r_3} + \vec{r_0} = \vec{0} \tag{2.9}$$

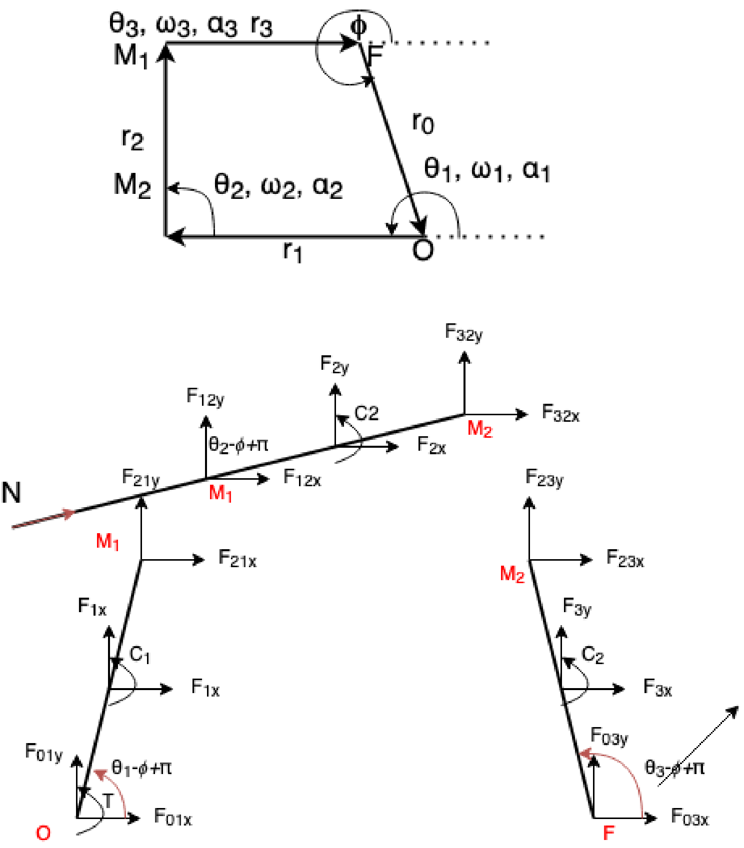
$$\vec{v_0} + \vec{v_1} + \vec{v_2} + \vec{v_3} = \vec{0} \tag{2.10}$$

$$\vec{a_0} + \vec{a_1} + \vec{a_2} + \vec{a_3} = \vec{0} \tag{2.11}$$

$$\vec{\omega_i} = [0, 0, \omega_{iz}]^\top \quad \forall \quad i \in \{1, 2, 3\} \tag{2.12}$$



(a) Model of a MePed Leg



(b) Four Bar Linkage in the MePed Knee

Figure 2.12: Model of a MePed Leg and the Four Bar Linkage at the Knee

$$\vec{\omega}_0 = [0,0,0]^\top \tag{2.13}$$

$$\vec{v}_0 = [0,0,0]^\top \tag{2.14}$$

$$\vec{v}_i = \vec{\omega}_i \times \vec{r}_i \quad \forall \quad i \in \{1,2,3\}, \tag{2.15}$$

$$\vec{v}_i = [-\omega_{iz} r_{iy}, \omega_{iz} r_{ix}, 0]^\top \quad \forall \quad i \in \{1,2,3\} \tag{2.16}$$

$$\omega_{2z} = -\omega_{1z} \frac{r_{1y} r_{3x} - r_{3y} r_{1x}}{r_{2y} r_{3x} - r_{3y} r_{2x}} \tag{2.17}$$

$$\omega_{3z} = -\omega_{1z} \frac{r_{2y} r_{1x} - r_{1y} r_{2x}}{r_{2y} r_{3x} - r_{3y} r_{2x}} \tag{2.18}$$

$$\vec{\alpha}_0 = [0,0,0]^\top \tag{2.19}$$

$$\vec{\alpha}_i = [0,0,\alpha_{iz}]^\top \quad \forall \quad i \in \{1,2,3\} \tag{2.20}$$

$$\vec{a}_0 = 0 \tag{2.21}$$

$$\vec{a}_i = \vec{\alpha}_i \times \vec{r}_i + \vec{\omega}_i \times \vec{\omega}_i \times \vec{r}_i \quad \forall \quad i \in \{1,2,3\} \tag{2.22}$$

$$\beta_3 = -\alpha_{1z} r_{1y} - \omega_{1z}^2 r_{1x} - \omega_{2z}^2 r_{2x} - \omega_{3z}^2 r_{3x} \tag{2.23}$$

$$\beta_4 = -\alpha_{1z} r_{1x} - \omega_{1z}^2 r_{1y} - \omega_{2z}^2 r_{2y} - \omega_{3z}^2 r_{3y} \tag{2.24}$$

$$\alpha_{2z} = \frac{r_{3x}\beta_3 - r_{3y}\beta_4}{r_{2y} r_{3x} - r_{3y} r_{2x}} \tag{2.25}$$

$$\beta_5 = -\alpha_{1z} r_{1y} - \omega_{1z}^2 r_{1x} - \omega_{2z}^2 r_{2x} - \omega_{3z}^2 r_{3x} \tag{2.26}$$

$$\beta_6 = -\alpha_{1z} r_{1x} - \omega_{1z}^2 r_{1y} - \omega_{2z}^2 r_{2y} - \omega_{3z}^2 r_{3y} \tag{2.27}$$

$$\alpha_{3z} = \frac{-r_{2x}\beta_5 + r_{2y}\beta_6}{r_{2y} r_{3x} - r_{3y} r_{2x}} \tag{2.28}$$

Since O is the driven joint in the knee, θ_1, ω_1 and α_1 are known and can be directly plugged into the equations above to obtain the kinematic parameters of the knee. Furthermore, the following equations determine the dynamics of the knee four bar linkage.

$$F_{01x} + F_{1x} + F_{21x} = 0 \tag{2.29}$$

$$F_{01y} + F_{1y} + F_{21y} = 0 \tag{2.30}$$

$$F_{01y} + F_{1y} + F_{21y} = 0 \tag{2.31}$$

$$F_{01x} + F_{1x} + F_{21x} = 0 \tag{2.32}$$

$$\vec{F}_{23} = -\vec{F}_{32} \tag{2.33}$$

$$\vec{F}_{21} = -\vec{F}_{12} \tag{2.34}$$

$$C_i = 0 \quad \forall \quad i \in \{1, 2, 3, \} \tag{2.35}$$

$$F_{12x} + F_{2x} + F_{32x} + N\cos\theta_2 - \phi + \pi = 0 \tag{2.36}$$

$$F_{12y} + F_{2y} + F_{32y} + N\sin\theta_2 - \phi + \pi = 0 \tag{2.37}$$

$$\begin{aligned} T + C_1 - F_{1x}\frac{|\vec{r_1}|}{2}\cos\theta_1 - \phi + \pi + F_{1y}\frac{|\vec{r_1}|}{2}\cos\theta_1 - \phi + \pi \\ +F_{21y}|\vec{r_1}|\cos\theta_1 - \phi + \pi - F_{21x}|\vec{r_1}|\sin\theta_1 - \phi + \pi = 0 \end{aligned} \tag{2.38}$$

$$\begin{aligned} C_2 - F_{2x}\frac{|\vec{r_2}|}{2}\cos\theta_2 - \phi + \pi + F_{2y}\frac{|\vec{r_2}|}{2}\cos\theta_2 - \phi + \pi \\ +F_{32y}|\vec{r_2}|\cos\theta_2 - \phi + \pi - F_{32x}|\vec{r_2}|\sin\theta_2 - \phi + \pi = 0 \end{aligned} \tag{2.39}$$

$$\begin{aligned} C_3 - F_{3x}\frac{|\vec{r_3}|}{2}\cos\theta_3 - \phi + \pi + F_{3y}\frac{|\vec{r_3}|}{2}\cos\theta_3 - \phi + \pi \\ -F_{23y}|\vec{r_1}|\cos\theta_1 - \phi + \pi + F_{23x}|\vec{r_1}|\sin\theta_1 - \phi + \pi = 0 \end{aligned} \tag{2.40}$$

2.4.2 Simplified Description Kinematics & Dynamics

The net force and torque acting on the quadruped are of more importance than the internal forces acting within the quadruped. The net force and torque are used for the quantification of the stability of quadruped gait. Equations 2.41 and 2.42 describe the overall dynamics of a quadruped.

$$\sum \vec{F}_{fr} + \sum \vec{N} - m\vec{g} = m\vec{a} \tag{2.41}$$

$$\sum_i \vec{F}_{fr} \times R_i + \sum_i \vec{N} \times R_i = I_R\vec{\alpha_R} \tag{2.42}$$

Gazebo and ROS were used for simulation of the simplified quadruped description. They provides a number of plugins for sensor integration into the simulations. Moreover, a number of kinematics and dynamics solvers are available that can use the integrated sensor information to calculate relevant kinematics and dynamics parameters for the quadruped. The following sensor plugins are currently integrated with the simulation.

1. IMU plugin
2. Gazebo Bumper Plugin

The following ROS packages are used for calculating the kinematics and dynamics of the quadruped.

1. MoveIt!
2. Orocos KDL solver
3. tf2

Equations 2.41 and 2.42 form the basis of the all the dynamics calculations performed by the three packages. *tf2* and *MoveIt* create a real-time kinematic model of a simulated robot to calculate end-effector positions and other kinematics parameters. Figure 2.13 depicts the chains that *tf2* generates for calculating coordinate transformations.

2.5 Summary

The simulation environment developed for the MePed and its simplified quadruped description serves as the training environment for RL with the DNN-CPG controller. Moreover, the environment has been developed for other generic quadruped simulation tasks such as path planning. The study of kinematics and dynamics of MePed form the basis for the formulation of the reward functions for quantification of stability of gait.

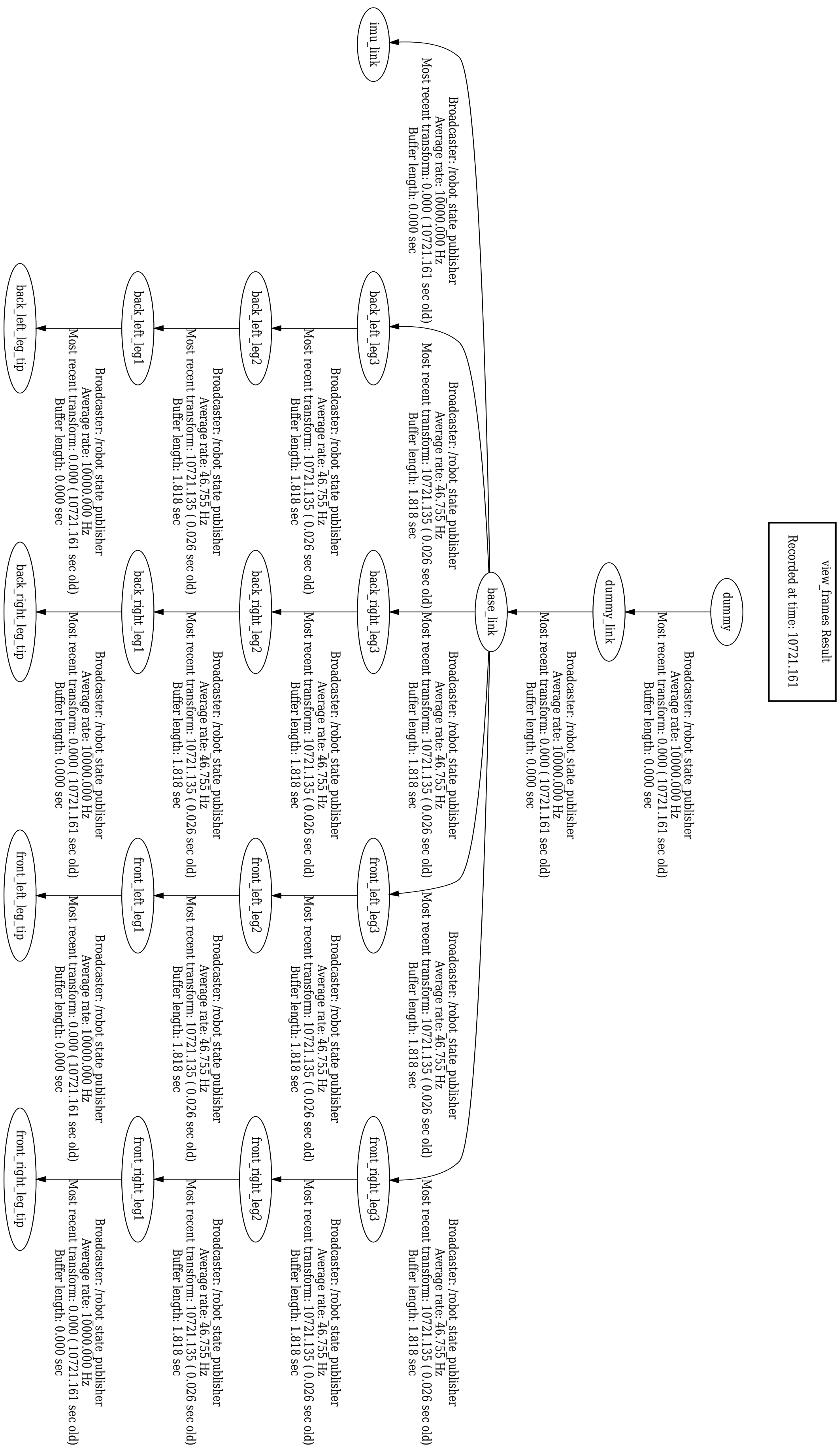


Figure 2.13: *tf2* tree of the simplified quadruped description

CHAPTER 3

GAIT LEARNING

3.1 Introduction

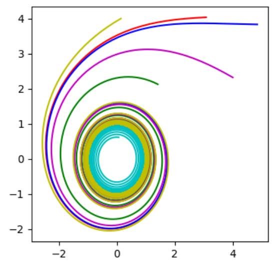
Being inspired by biology, recently researchers have proposed CPGs to generate policies for locomotion of robots ([2], [14], [15], [3]). However, the modulation of gait produced by such policies requires high dimensional parameter inputs, unlike in biology, where a low dimensional tonic signal is used. Moreover, learning to produce various gait for challenging environments is difficult, and there is no standard methodology. On the other hand, DNNs enjoy standard learning techniques and the ability of universal approximation. Thus, a combination of DNNs and CPGs could be used to generate gait for robots that could be easily modulated with low dimensional inputs or feedback. The motivation for developing a combined DNN-CPG model was established in detail in Section 1.2. This Chapter describes the building blocks of the DNN-CPG architecture and its construction.

3.1.1 Hopf Oscillator Model

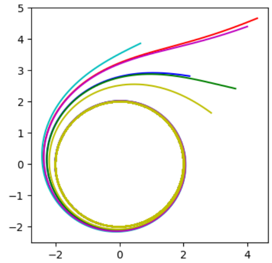
Various types of oscillator models have been robotics so far. Though such oscillators are capable of generating motion trajectories for robots, the relationship between the output variables and the model parameters is complex and hard to understand. Hopf Oscillators, on the other hand, have a simple relationship between the model parameters and the output variables, and they exhibit limit cycle and synchronization behaviour [19]. This makes Hopf Oscillator for modelling the rhythmic motion trajectories for robots. The mathematical model of the Hopf oscillator is represented as follows,

$$\dot{z} = (\mu - \|z\|^2)z + \iota\omega z, \tag{3.1}$$

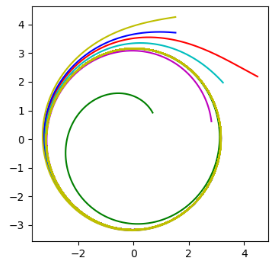
where z is a complex number. The same model can also be represented in the Cartesian coordinate system as follows,



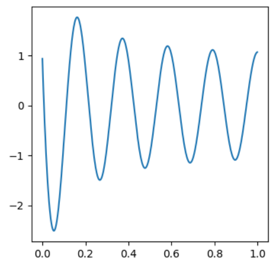
(a) Phase Plane of $\mu = 1$ and $\omega = 30$



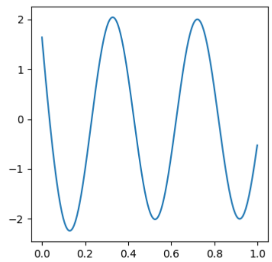
(b) Phase Plane of $\mu = 4$ and $\omega = 16$



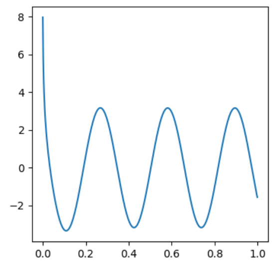
(c) Phase Plane of $\mu = 10$ and $\omega = 20$



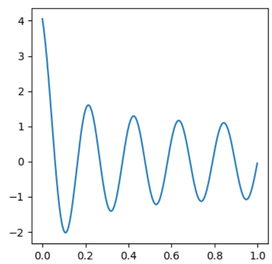
(d) Oscillations of real part of oscillator variable with $\mu = 1$ and $\omega = 30$



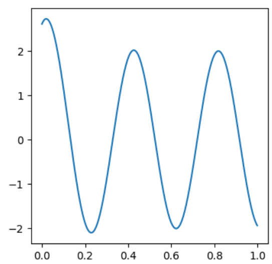
(e) Oscillations of real part of oscillator variable with $\mu = 4$ and $\omega = 16$



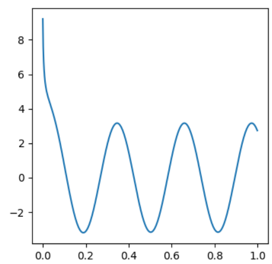
(f) Oscillations of real part of oscillator variable with $\mu = 10$ and $\omega = 20$



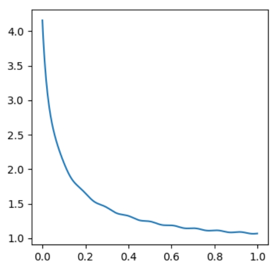
(g) Oscillations of imaginary part of oscillator variable with $\mu = 1$ and $\omega = 30$



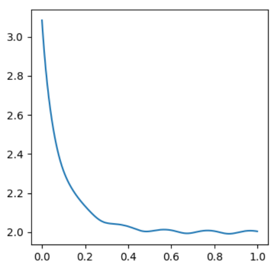
(h) Oscillations of imaginary part of oscillator variable with $\mu = 4$ and $\omega = 16$



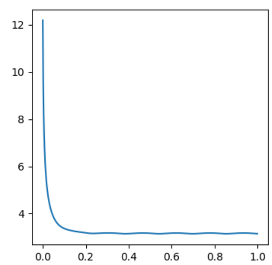
(i) Oscillations of imaginary part of oscillator variable with $\mu = 10$ and $\omega = 20$



(j) Amplitude of Oscillation for $\mu = 1$ and $\omega = 30$



(k) Amplitude of Oscillation for $\mu = 4$ and $\omega = 16$



(l) Amplitude of Oscillation for $\mu = 10$ and $\omega = 20$

Figure 3.1: Behaviour of the Hopf Oscillator with different parameter values.

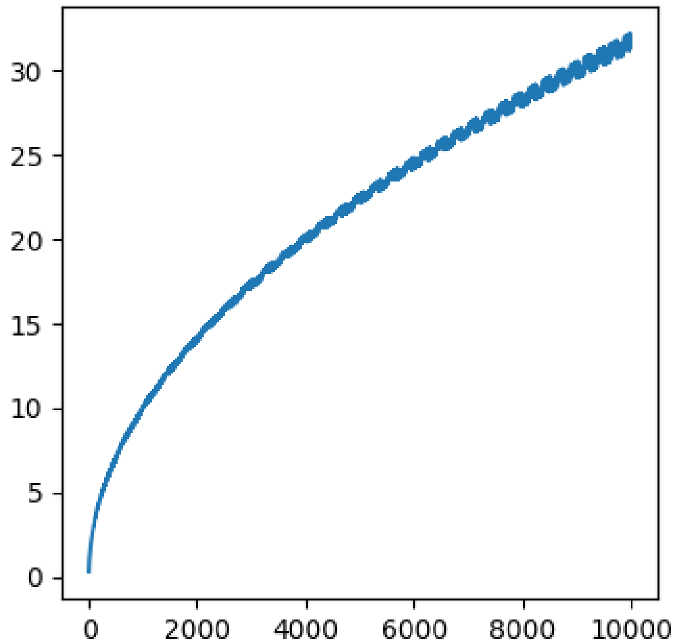


Figure 3.2: Trend of steady state amplitude of oscillations with oscillator parameter μ

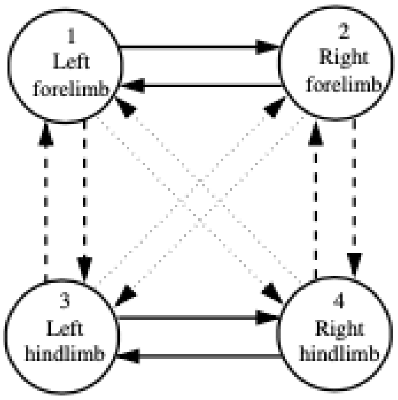
$$\dot{x} = -\omega y + x(\mu - x^2 - y^2) \tag{3.2}$$

$$\dot{y} = \omega x + y(\mu - x^2 - y^2) \tag{3.3}$$

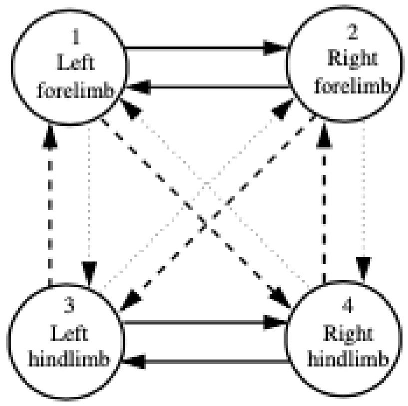
Figure **??** depicts the behaviour of the hopf oscillator for different values of its parameters μ and ω The parameter μ controls the amplitude of steady-state oscillations, and the parameter ω controls the frequency of steady-state oscillations. The amplitude of oscillations settles to a steady-state value of $\sqrt{\mu}$. Figure 3.2 depicts the trend of amplitude of oscillations at steady state vs μ.

3.1.2 Existing CPG Architecture

As previously mentioned, a number of CPG architecture have been successful in producing rhythmic patterns for generating motion trajectories. A number of such architectures for quadrupeds are successive improvements upon the fully connected network of oscillators proposed in [3]. Figure 3.3 describes the coupling structure of the 4 cell network from [3]. Such networks take advantage of the synchronization between oscillators for produce adaptive rhythmic patterns. This model was further extended in [16] by introducing a two-level hierarchy, such that the higher level modulates the CPG



(a) Trot Pace Bound Net



(b) Walk Net.

Figure 3.3: Generic coupling structure for a 4 cell network in [3]

model by appropriately modifying CPG parameters. Another architecture of interest is the CPG model, proposed in [4], driving a salamander robot. The CPG model is based on the CPGs observed in a lamprey and has been successfully used to show transition from terrestrial behaviour to aquatic behaviour. Moreover, the motion trajectory generation by this model is controlled by a low-dimensional tonic signal. However, the coupling weights of the CPG are hand tuned and learning with this architecture is extremely difficult. Figure 3.4 depicts the CPG model in [4].

A departure from the fully-connected oscillator network architecture was proposed in [5]. CPG-based methods seek to utilise the ability of CPGs to entrain with the body dynamics of the robot. A high-level controller was introduced in [5], modulating the CPG network and leading to greater generalisation abilities and a more effective form of control. The high-level controller is implemented as a fully-connected, feed-forward neural network with two hidden layers containing 400 and 300 ReLU units, respectively and is trained to minimise the lateral deviation through RL. Figure 3.5 depicts the detailed DNN-CPG architecture for stable bipedal gait. The DNN-CPG proposed in this work builds upon the architecture in Figure 3.5 and attempts to integrate the CPG into the DNN for simultaneous optimisation of parameters.

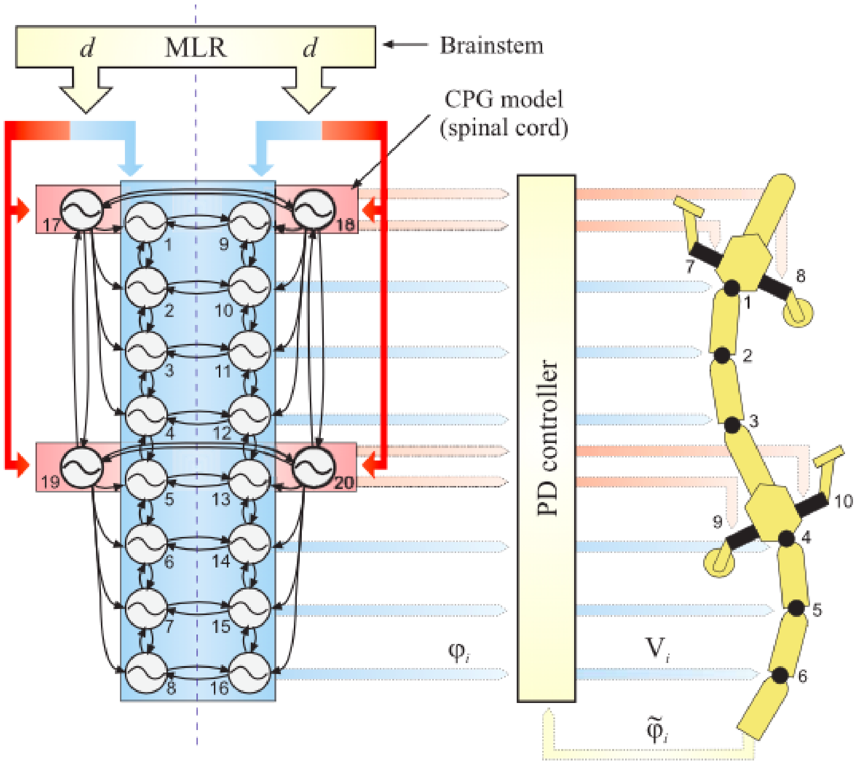


Figure 3.4: Configuration of the CPG driving the Salamander Robot in [4]

3.2 DNN-CPG

A DNN-CPG architecture to generate rhythmic motion trajectories for robotic control applications must generate rhythms, interpret a low dimensional modulating input, and appropriately modulate the generated rhythms. In such an architecture, the CPG performs rhythm generation, whereas the DNN interprets inputs and appropriately modulates the CPG parameters to modulate generated rhythm. For instance, in the DNN-CPG architecture proposed in [5], the higher-level controller outputs two parameters ψ_l and ψ_r. These parameters then directly modify the amplitude of the hip joint rhythms to maintain balance.

3.2.1 Fourier Decomposition of Signals

A periodic signal may be represented as a (possibly infinite) weighted summation of harmonically related sinusoids. For instance, a square wave may be represented as a sum of sinusoids to varying accuracy. Figure 3.6 depicts reconstruction of a square wave from its fourier components.

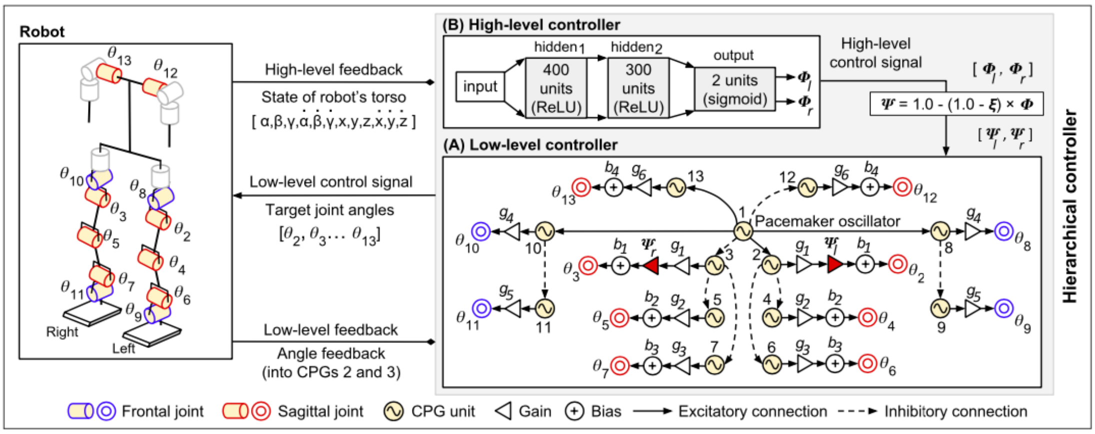


Figure 3.5: Detailed DNN-CPG architecture proposed in [5]

The limit cycle exhibited by the Hopf Oscillator at steady state can be leveraged to perform fourier reconstruction of a periodic signal, given that the fourier coefficients for the periodic signal are known. The following equation depicts fourier decomposition of a signal.

$$s(t) = \sum_{k=-\infty}^{\infty} C_k e^{\iota k \omega_o t} \tag{3.4}$$

The Hopf Oscillator, being a complex exponential at steady state, can be used as the Fourier Reconstruction basis. Thus, the Fourier Decomposition can be rewritten as follows,

$$s(t) = \sum_{k=-\infty}^{\infty} C_k z_k(t) \tag{3.5}$$

where $z_k(t)$ is the dependent variable of a Hopf Oscillator with frequency of oscillations (ω) equal to $k\omega_o$ and ω_o is the fundamental frequency of the periodic signal $s(t)$. Thus, leveraging the limit cycle behaviour of the Hopf Oscillator, n Hopf Oscillators, each with a frequency, an integer multiple of ω_0 can be used to approximately reconstruct $s(t)$. This is similar to how the Fourier components of the square waves were used to reconstruct it to varying degrees of accuracy in Figure 3.6.

3.2.2 Motion Trajectory Generation

So far, we have established that a collection of Hopf Oscillators can be used to reconstruct any periodic signal to some degree of accuracy, given that the Fourier coefficients are known. The property of universal approximation of neural networks can be utilised

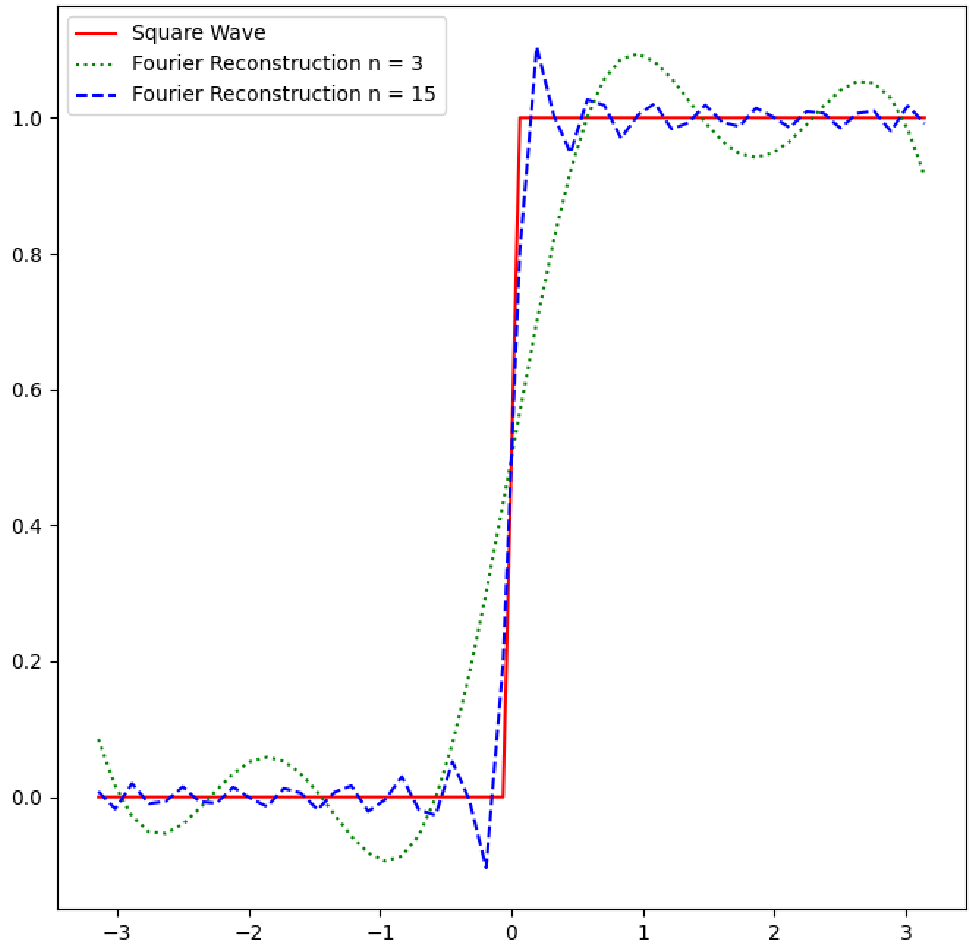


Figure 3.6: Fourier Series Reconstruction of a Square Wave with n fourier components

to combine the simple harmonics from the Hopf Oscillator and produce rhythmic motion trajectories. Figure 3.7 depicts the neural network used to generate motion trajectories for particular values of ω and μ.

The Complex MLP takes the harmonics produced by the CPG, produces the appropriate coefficients, multiplies the harmonics, and adds the scaled harmonics to produce motion trajectories. The neural network depicted in 3.7 produces, for instance, the motion trajectories for the 8 DoF in the MePed. The network consists of a single layer of Hopf Oscillators, which constitute the CPG and several fully connected complex-valued dense layers connected in a feed-forward fashion.

3.2.3 Motion Trajectory Modulation

The simple DNN-CPG architecture depicted in Figure 3.7 can generate rhythmic motion trajectories but has no means of modulation or interpretation of inputs. Moreover,

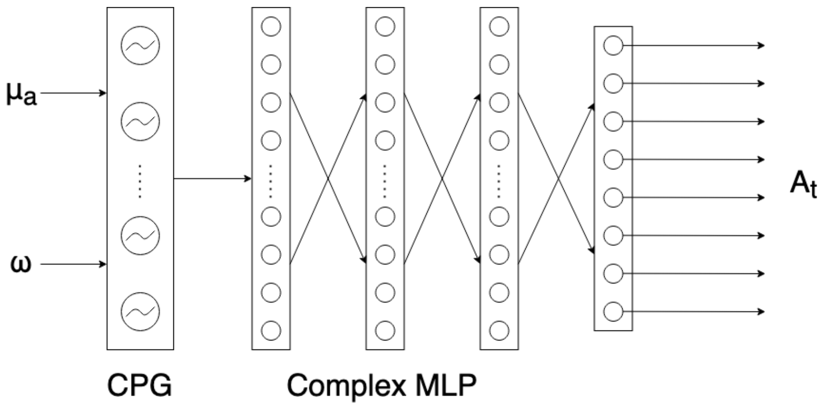


Figure 3.7: A Simple DNN-CPG Architecture for generating rhythmic motion trajectories given ω and μ

any modulation of the needs to performed by directly modifying CPG parameters ω and μ_a. Thus, a motion encoding network is used to interpret desired motion trajectory inputs and output appropriate ω and μ. The motion encoding network consists of several fully connected feed-forward layers. The desired motion trajectory inputs is a concatenation of the following two vectors:

1. A Heading Vector
2. A Desired Robot Velocity Vector

A robot state encoding network is used to facilitate the interpretation of feedback from sensors. The robot state encoding network is also a multi-layered, fully connected feed-forward network. It takes the robot state as input. The robot state is a concatenation of the following vectors:

1. Current Joint Angles
2. Difference between current and last joint positions
3. Robot Orientation Vector
4. Robot Angular Velocity Vector
5. Robot Linear Acceleration Vector

Another network of several fully connected and convolutional layers will also be introduced to emulate more sophisticated behaviour such as obstacle negotiation and path planning. The convolutional network will take as input the images from a camera mounted on the robot. The outputs of the motion encoding network, the robot state encoding network and the aforementioned convolutional network will be concatenated and directly passed onto the Complex MLP. Such an architecture would facilitate motion trajectory modulation without disturbing motion trajectory generation. Figure 3.8 depicts the described DNN-CPG architecture.

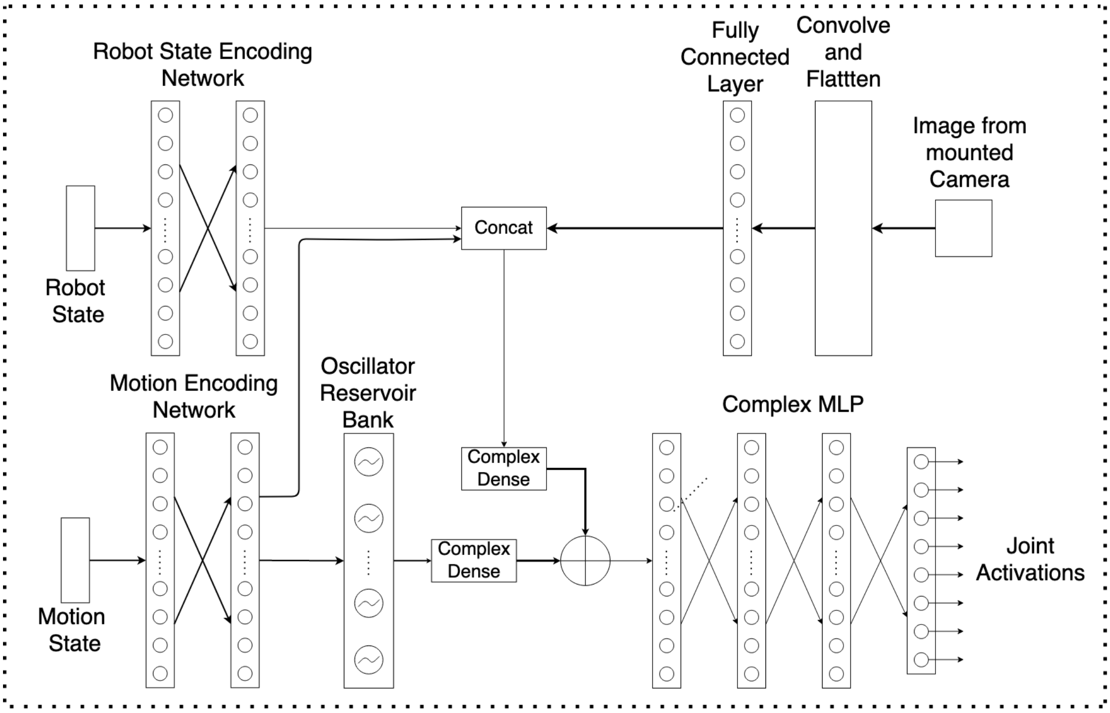


Figure 3.8: DNN-CPG architecture to ensure stable gait.

Each component of the CNN-CPG architecture in 3.8 may be mapped to a part of the nervous system. For instance, the CPG and the complex MLP constitute the spinal cord. The robot state encoding network constitutes the proprioceptive pathways; the convolutional neural network constitutes the visual pathways, and the motion encoding network constitutes the pedunculopontine nucleus in the lower brain stem, whose outputs drives the CPG.

3.2.4 Learning

An incremental approach to learning behaviour is followed. The modular architecture of the DNN-CPG allows for integrating modules for the emulation of more sophisticated behaviour in parallel to the existing architecture. This allows for a multi-loop hierarchical control system. The parallel computation that such a control system executes is very similar to how the brain operates. Thus, some of the characteristics that are expected to develop naturally in the control system are as follows:

1. Multiple Goal fulfilment

2. Multiple Sensor Information Fusion
3. Additivity of sophisticated behaviour

Figure 3.9 depict a decomposition of the proposed control system. In this work, both supervised learning through DL and unsupervised learning through RL are used. Supervised learning is used to provide a robust initialization to the DNN-CPG for quick convergence through RL.

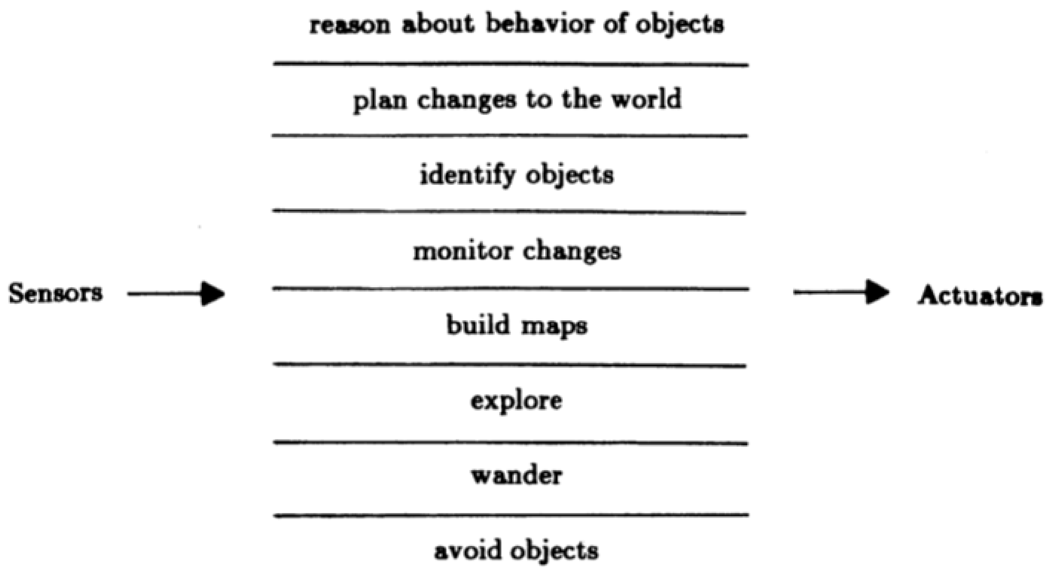


Figure 3.9: A Decomposition of a mobile robot control system based on task achieving behaviours [6]

3.3 Supervised Learning

Supervised Learning is used for the initialisation of DNN-CPG for RL. It is also used to prove the ability of the network to perform as expected. Ideal Motion Trajectory and its relationship with the speed of the quadruped were formulated. Supervised Learning was then performed in two steps. First, the simple DNN-CPG in Figure 3.7 was trained, then the entire architecture in Figure 3.8 (without the visual pathway) was trained.

3.3.1 Ideal Motion Trajectory

Inspired by the quadruped mammals' gaits, many different methods have been proposed to design a gait for the quadruped robot's locomotion, like drive function and foot trajectory. The walk gait design based on sinusoidal drive functions, proposed in [20], is used to synthesise data for supervised learning. The following equations describe the motion trajectories for the driven joints of the MePed. The trajectories may be divided into a stance phase and a swing phase. A leg is in the stance phase when in contact with the ground and pushing the robot in the direction of motion, and the swing phase during the leg's motion when not in contact with the ground. Figure 3.10 depicts the construction of the drive function for a leg. The drive function is a piece-wise sinusoidal, consisting of a stance phase of lower frequency and a swing phase of a higher frequency. Figure 3.11 depicts the motion trajectories generated from the drive function. The joint trajectory for the ankle joint is the same as the joint trajectory for the knee joint scaled by -0.01. Refer to Table 3.1 for the leg index to leg name mapping used in the following equations.

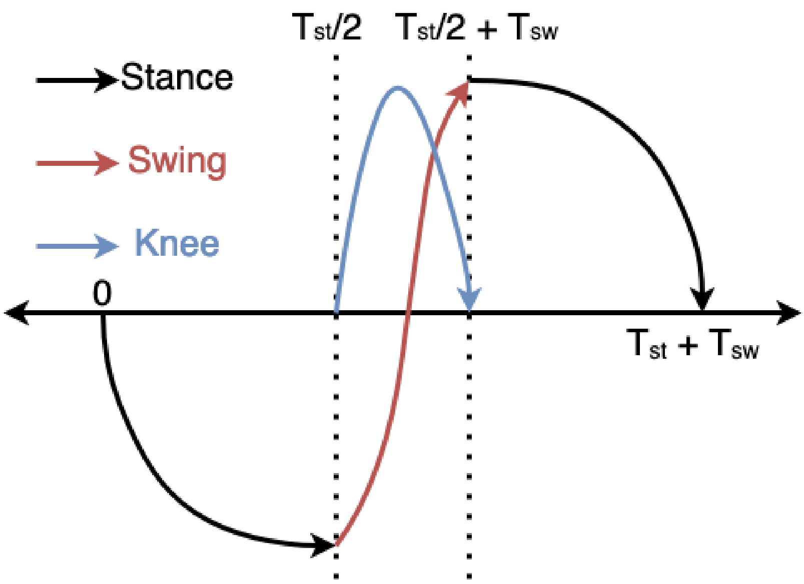


Figure 3.10: Construction of the Drive function for a leg

Table 3.1: Leg Index to Name mapping

Index	Leg Name
0	Front Right Leg
1	Back Right Leg
2	Front Left Leg
3	Back Left Leg

$$T = T_{sw} + T_{st} \tag{3.6}$$

$$\beta = \frac{T_{st}}{T_{st} + T_{sw}} \tag{3.7}$$

$$\theta_h(t) = \begin{cases} \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \pi), & \text{if } 0 \leq t \leq \frac{\beta T}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{(1-\beta)T} + \frac{(3-4\beta)\pi}{2(1-\beta)}), & \text{if } \frac{\beta T}{2} \leq t \leq \frac{T(2-\beta)}{2} \\ \theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \frac{(\beta-1)\pi}{\beta}), & \text{if } \frac{T(2-\beta)}{2} \leq t \leq T \end{cases} \quad \forall \quad i \in \{0,1\} \tag{3.8}$$

$$\theta_h(t) = \begin{cases} -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \pi), & \text{if } 0 \leq t \leq \frac{\beta T}{2} \\ -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{(1-\beta)T} + \frac{(3-4\beta)\pi}{2(1-\beta)}), & \text{if } \frac{\beta T}{2} \leq t \leq \frac{T(2-\beta)}{2} \\ -\theta_h sin(\frac{(t-\frac{iT}{4})\pi}{\beta T} + \frac{(\beta-1)\pi}{\beta}), & \text{if } \frac{T(2-\beta)}{2} \leq t \leq T \end{cases} \quad \forall \quad i \in \{2,3\} \tag{3.9}$$

$$\theta_k(t) = \begin{cases} \theta_k sin(\frac{t\pi}{T(1-\beta)} - \frac{\beta\pi}{2(1-\beta)}), & \text{if } \dot{\theta_h}(t) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad \forall \quad i \in \{0,1\} \tag{3.10}$$

$$\theta_k(t) = \begin{cases} \theta_k sin(\frac{t\pi}{T(1-\beta)} - \frac{\beta\pi}{2(1-\beta)}), & \text{if } \dot{\theta_h}(t) \leq 0 \\ 0, & \text{otherwise} \end{cases} \quad \forall \quad i \in \{2,3\} \tag{3.11}$$

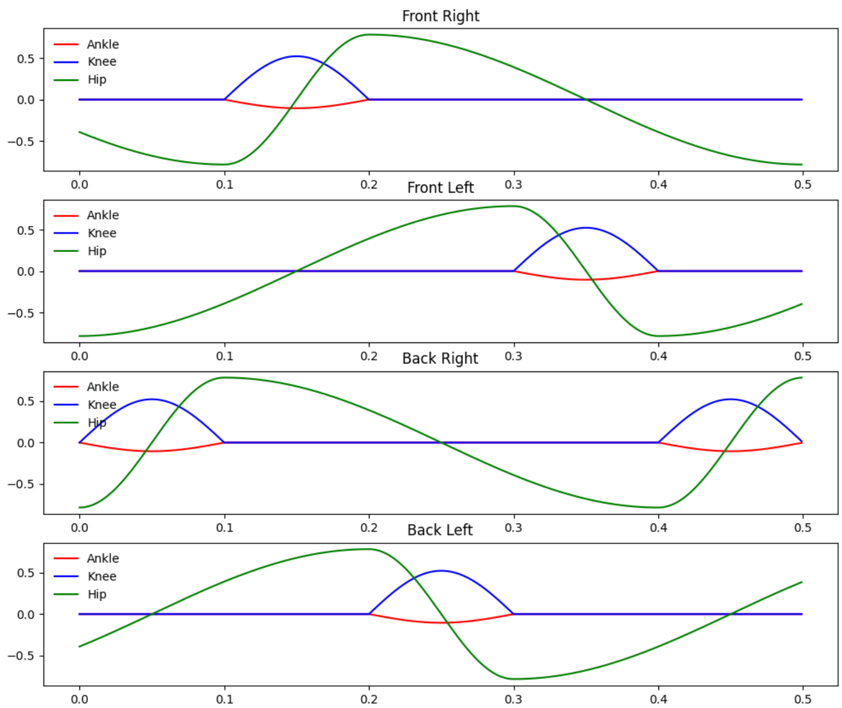


Figure 3.11: Assumed ideal gait pattern for quadruped locomotion

For an ideal motion trajectory, the following relationships hold true

1. $v \propto \frac{1}{T_{sw}}$
2. $v \propto \frac{1}{T_{st}}$
3. $v \propto \theta_h$

These relationships are evident from a superficial analysis of hip joint movement and its impact on the robot movement. If the joint moves by an angle $2\theta_h$ from its initial to the final position and then back, the leg sweeps over a distance of $2L_T\theta_h$. This sweep happens over a time-period of T, the period of the gait cycle. This analysis can be used to compute the average speed of movement of the robot. Since the foot remains at a fixed position during the stance phase, the distance $2L_T\theta_h$ is covered by the robot in the direction of motion. Thus,

$$v = \frac{2L_T\theta_h}{T_{st} + T_{sw}} \tag{3.12}$$

A similar analysis was performed in [20] to determine the distance moved by the robot during one gait cycle.

3.3.2 Training Results

Supervised training of the DNN-CPG provided experimental proof that the architecture can successfully generate motion trajectories for a robot. First, the simple DNN-CPG architecture in Figure 3.7 was used trained using the fundamental frequency of the ideal rhythmic trajectories as input, then the DNN-CPG architecture depicted in Figure 3.8 was trained using the desired motion vector and current robot state as input. The fundamental frequency was calculated using the Fourier Transform of the motion trajectories of one of the hip joint. The ideal motion trajectories defined in Section 3.3.1 were used as the ground truth or labels in both cases. Table 3.2 maps neural network training experiments to their training error and generated output motion trajectories.

The training results of the simple DNN-CPG shows that the architecture can generate valid motion trajectories given a low dimensional input. However, the simple DNN-CPG can only produce motion trajectories for a straight-line motion, without any possibility of modulation of the trajectories.

The DNN-CPG architecture in Figure 3.8 provides the possibilities for trajectory modulation. This architecture is trained through RL. However, to initialise the network's parameters with meaningful values, this architecture was also trained by supervision.

Table 3.2: Training Results for the proof of concept simple DNN-CPG network

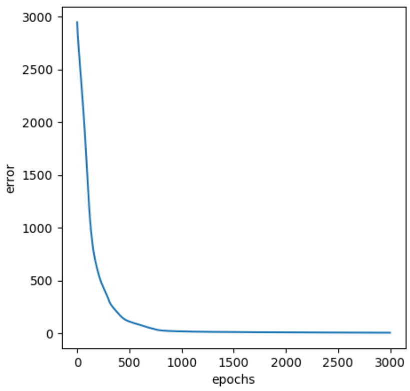
ID	Timesteps	LR	dt	T_{st} (ms)	T_{sw} (ms)	Osc.	Hidden Layer Neurons	Plots
1	500	0.001	0.001	60	20	20	50	Fig. 3.12a, Fig. 3.14a, Fig. 3.14b
2	500	0.001	0.001	60	20	40	200	Fig. 3.12b, Fig. 3.15a, Fig. 3.15b
3	500	0.001	0.001	60	20	20	200	Fig. 3.12c, Fig. 3.16a, Fig. 3.16b
4	400, 530, 665, 800, 930, 500	0.001	0.001	60, 100, 80, 120, 140, 75	20, 33, 26, 40, 46, 25	8	16	Fig. 3.12d, Fig. 3.17a, Fig. 3.17b

Four supervised learning procedures were experimented with to obtain proper convergence.

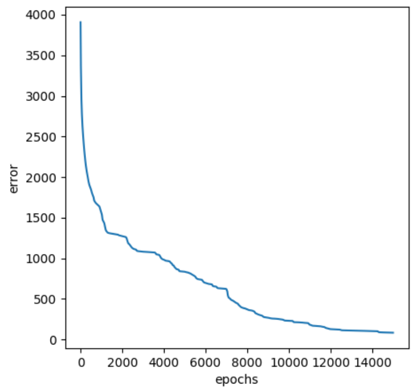
1. Training of the entire network in one go using MSE of ω, μ and generated motion trajectories
2. Training of the motion encoding network using MSE of ω and μ followed by training of entire network using MSE of ω, μ and generated motion trajectories
3. Training of the motion encoding network using MSE of ω and μ simultaneously with the entire network using MSE of generated motion trajectories
4. Training of the motion encoding network using MSE of ω and μ simultaneously with the training of complex MLP using MSE of generated motion trajectories

Training in two steps through procedure 2 proved more successful than other tested training procedures. The loss used in all experiments was Mean Squared Error. The following is the list of experiments performed:

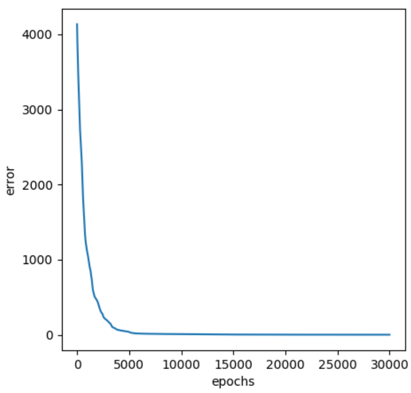
1. **Experiment 5** Training through procedure 3 with a learning rate of 0.1 decaying exponentially at a rate of 0.95 every 20 steps. Refer to Figure 3.13a
2. **Experiment 6** Training through procedure 4 with a learning rate of 0.1 decaying exponentially at a rate of 0.95 every 20 steps. Refer to Figure 3.13b
3. **Experiment 7** Training through procedure 1 with a constant learning rate of 0.01. Refer to Figure 3.13c.
4. **Experiment 8** Training through procedure 3 with a constant learning rate of 0.01. Refer to Figure 3.13d.



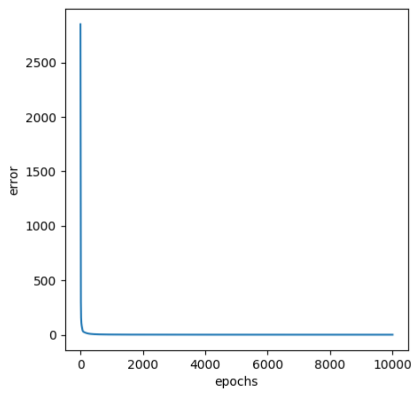
(a) Training Error Experiment ID 1



(b) Training Error Experiment ID 2



(c) Training Error Experiment ID 3



(d) Training Error Experiment ID 4

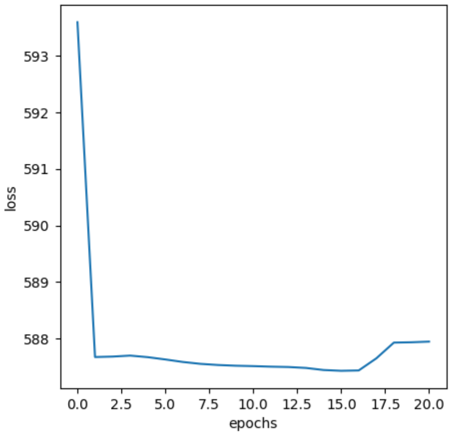
Figure 3.12: Training Error Plots for DNN-CPG architecture in Fig 3.7

5. **Experiment 9** Training through procedure 2 with a learning rate of 0.01 decaying exponentially at a rate of 0.95 every 20 steps. Refer to Figure 3.18e.
6. **Experiment 10** Training through procedure 2 with a constant learning rate of 0.01. Refer to Figure 3.19e.

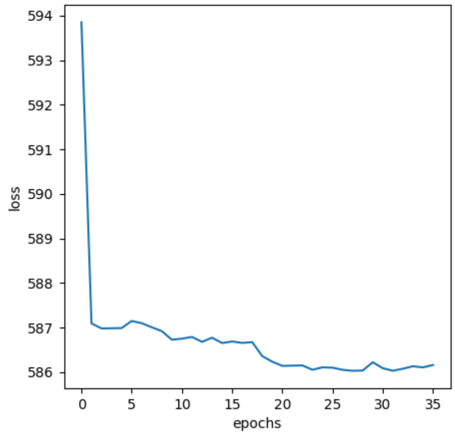
The model obtained after Experiment 9 was chosen for further RL due to the minimum training loss value at convergence.

3.4 Summary

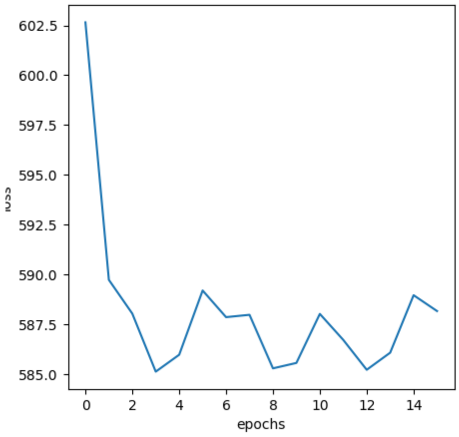
Supervised Learning serves as proof that the proposed architecture can generate appropriate motion trajectories given the desired motion input. Moreover, the "pre-training"



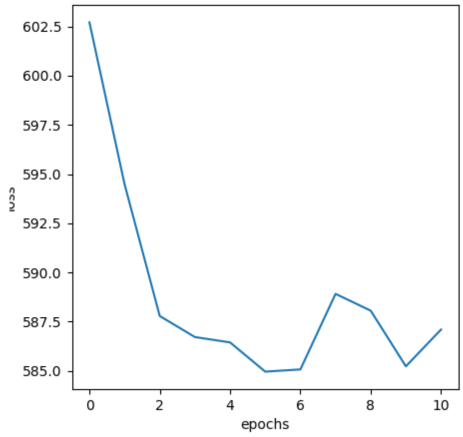
(a) Training Loss Experiment ID 5



(b) Training Loss Experiment ID 6



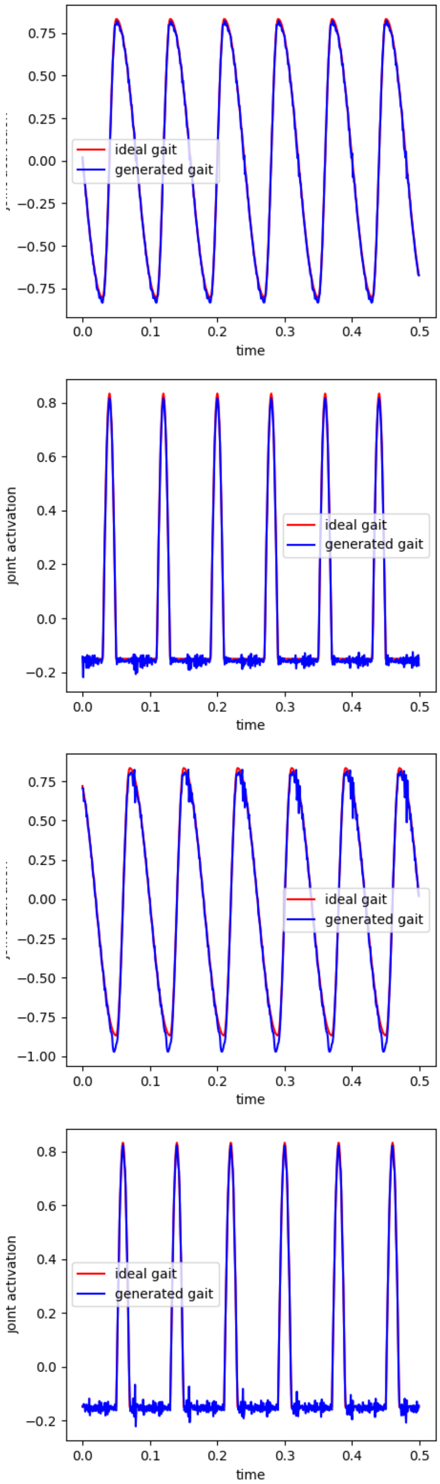
(c) Training Loss Experiment ID 7



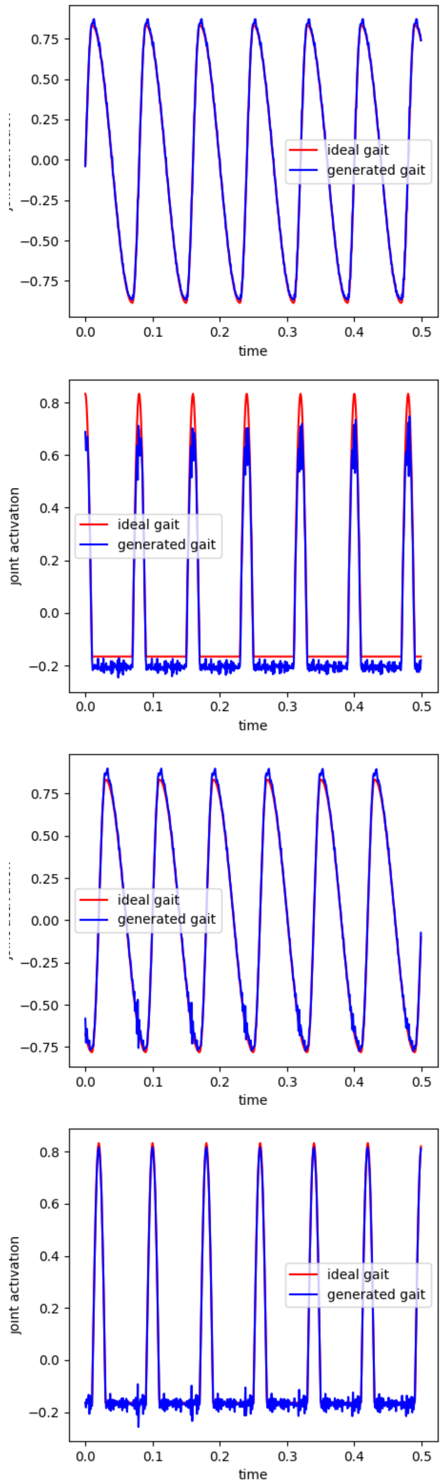
(d) Training Loss Experiment ID 8

Figure 3.13: Training Error Plots for entire DNN-CPG architecture in Figure 3.8 for Experiments 5-8

of the DNN-CPG will allow for easier convergence of the DDPG as the actor's weights (the DNN-CPG) are initialised to a meaningful value. This training procedure is very similar to how newly born animals know locomotion, though their abilities require fine-tuning experience. Similarly, the DNN-CPG "pre-trained" through supervised learning has the required knowledge for locomotion, though optimisation for a more stable gait is required.

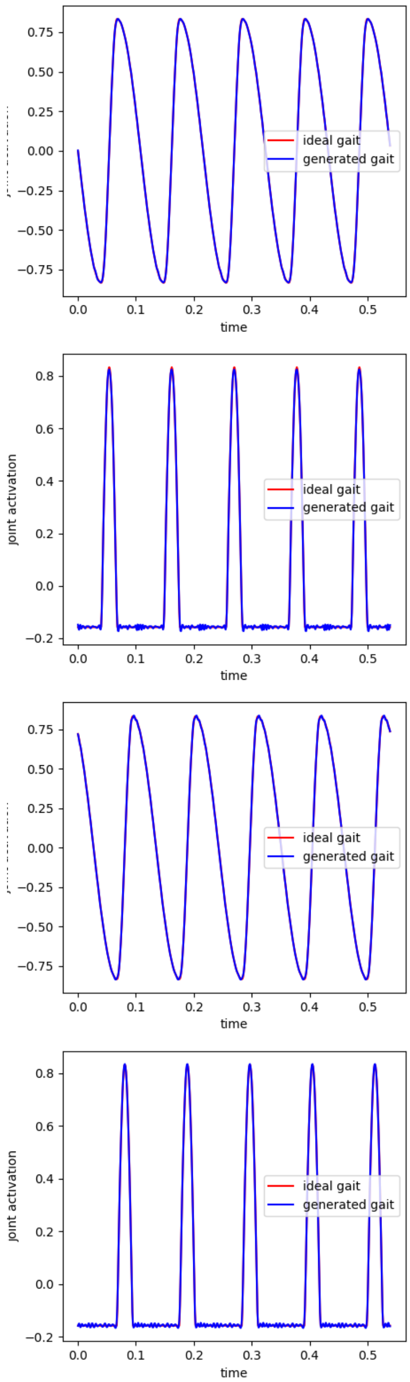


(a) Right Legs Joint Motion Trajectory

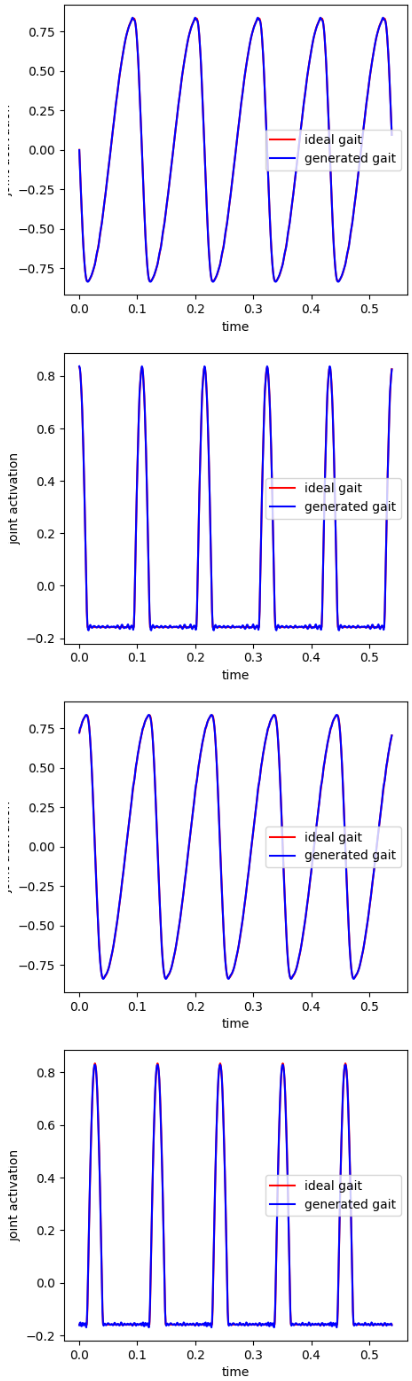


(b) Left Legs Joint Motion Trajectory

Figure 3.14: Experiment ID 1 Output Motion Trajectories

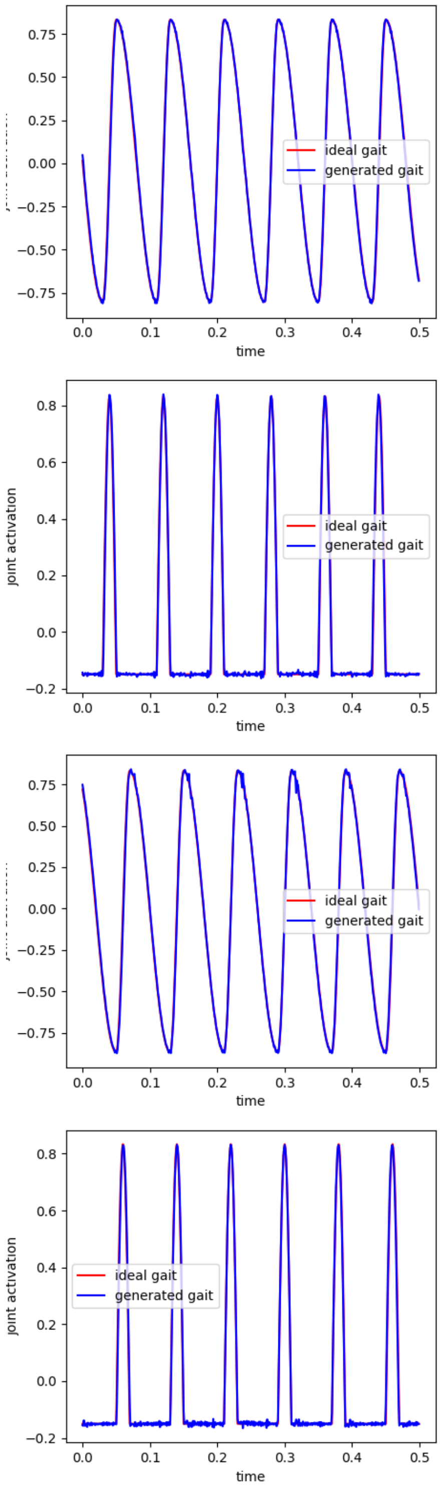


(a) Right Legs Joint Motion Trajectory

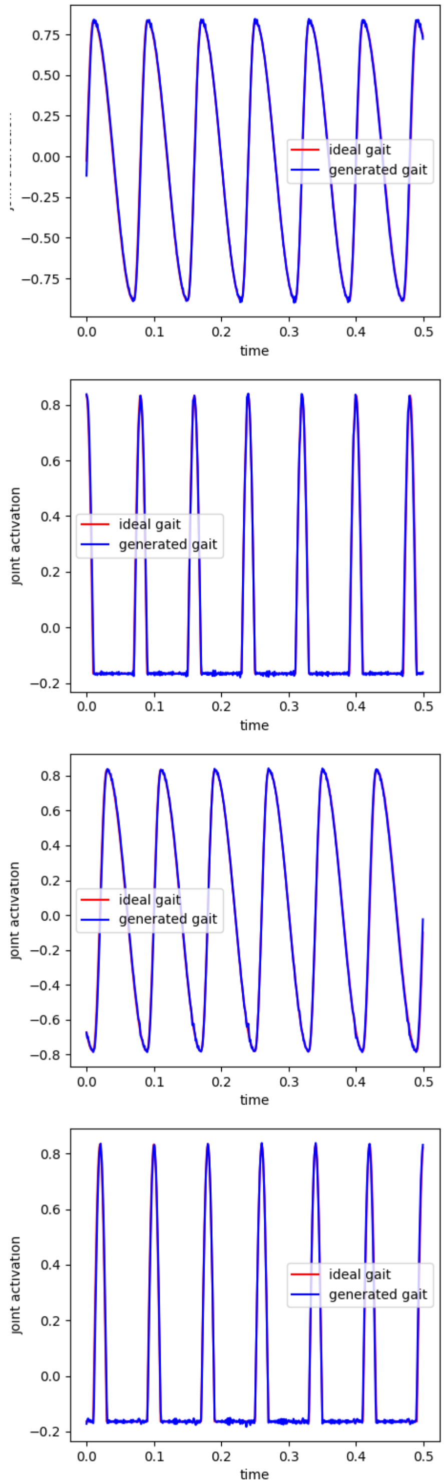


(b) Left Legs Joint Motion Trajectory

Figure 3.15: Experiment ID 2 Output Motion Trajectories

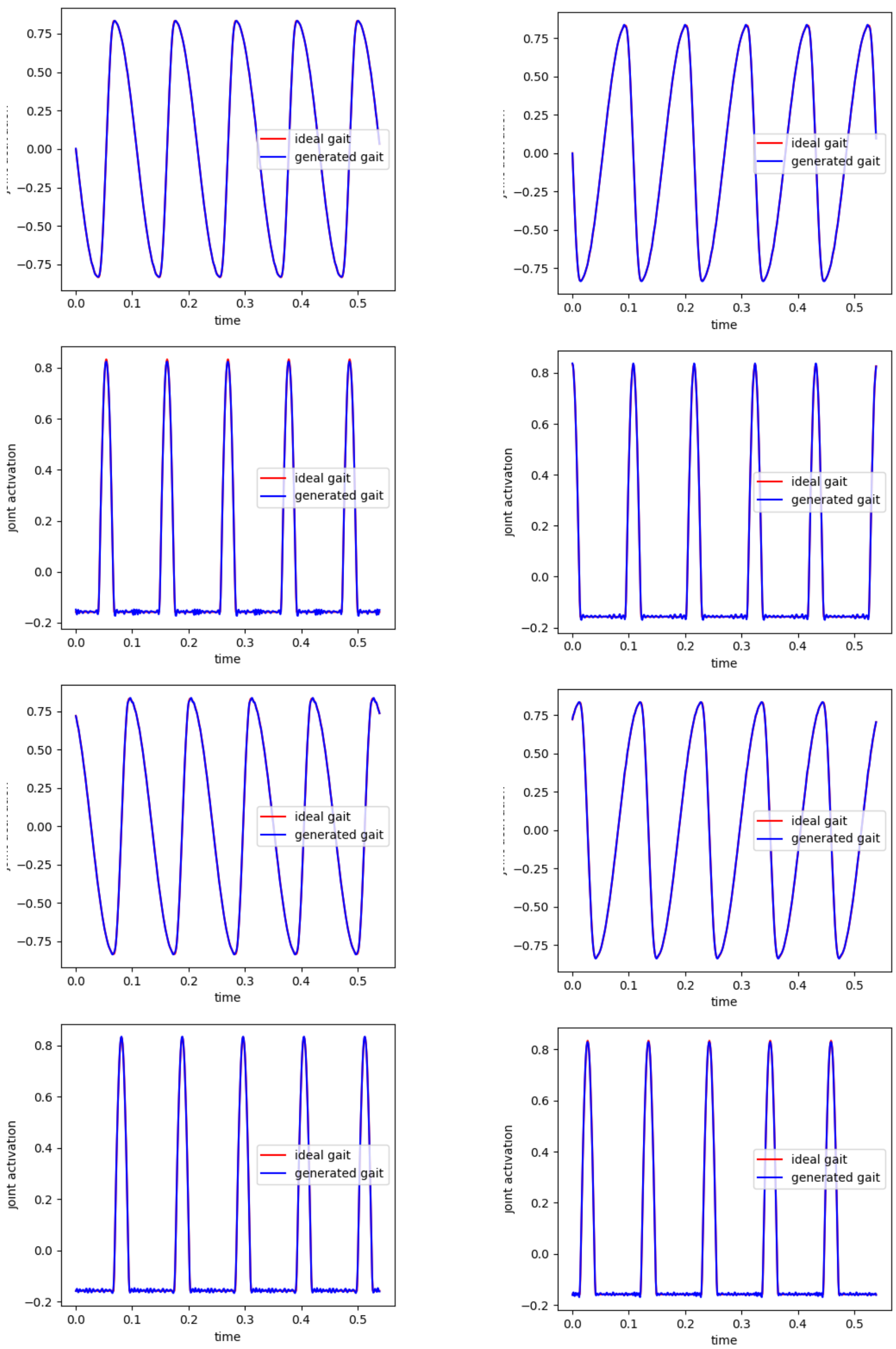


(a) Right Legs Joint Motion Trajectory



(b) Left Legs Joint Motion Trajectory

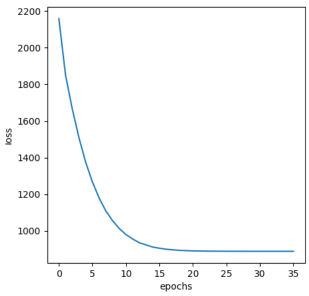
Figure 3.16: Experiment ID 3 Output Motion Trajectories



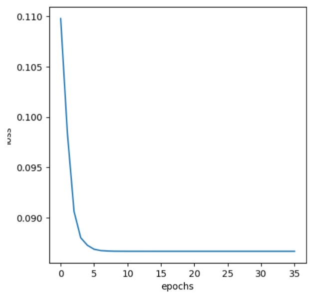
(a) Right Legs Joint Motion Trajectory

(b) Left Legs Joint Motion Trajectory

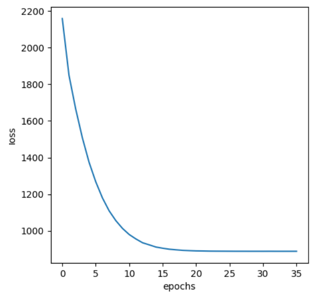
Figure 3.17: Experiment ID 4 Output Motion Trajectories



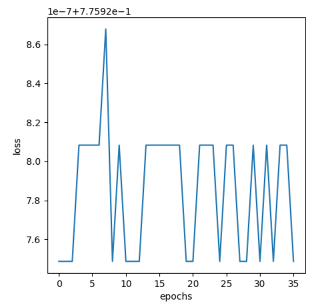
(a) Motion State Encoding Training Total Loss



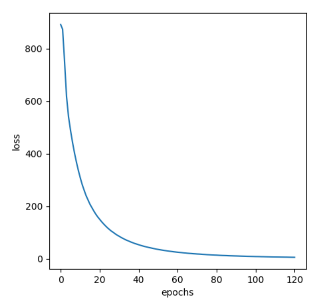
(b) Motion State Encoding Training μ Loss



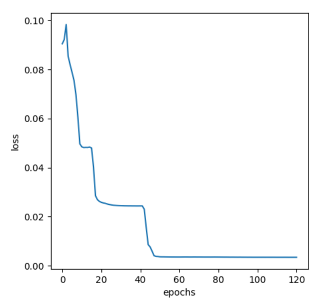
(c) Motion State Encoding Training ω Loss



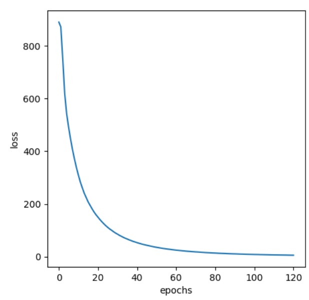
(d) Motion State Encoding Training Motion Trajectory Loss



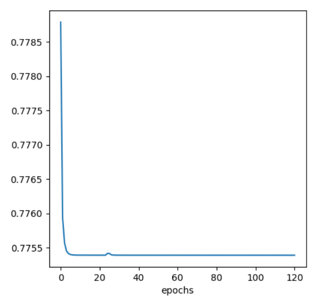
(e) Complete Network Training Total Loss



(f) Complete Network Training μ Loss

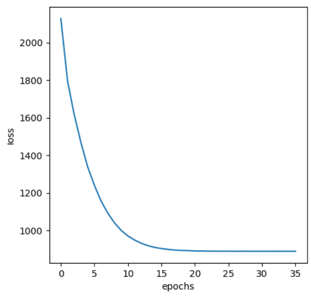


(g) Complete Network Training ω Loss

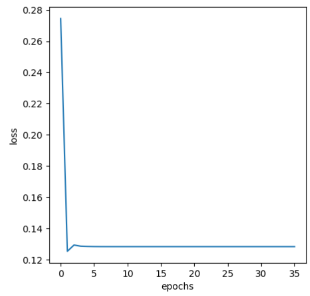


(h) Complete Network Training Motion Trajectory Loss

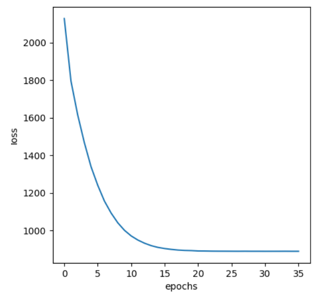
Figure 3.18: Training Error Plots for DNN-CPG architecture in two steps, experiment 9



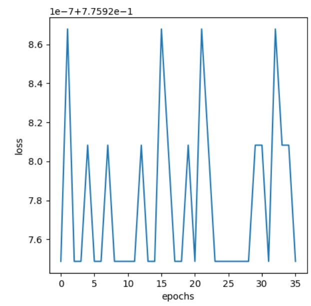
(a) Motion State Encoding Training Total Loss



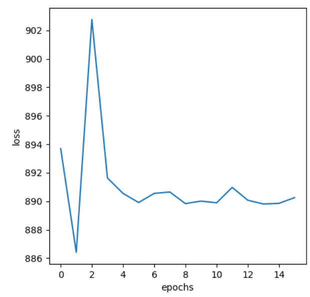
(b) Motion State Encoding Training μ Loss



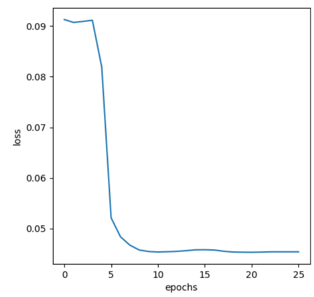
(c) Motion State Encoding Training ω Loss



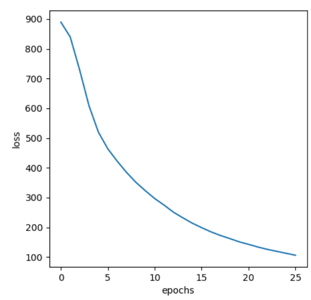
(d) Motion State Encoding Training Motion Trajectory Loss



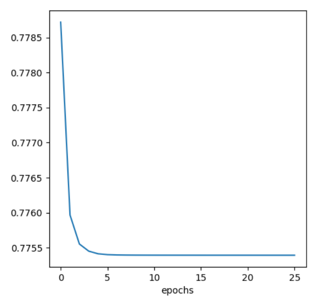
(e) Complete Network Training Total Loss



(f) Complete Network Training μ Loss



(g) Complete Network Training ω Loss



(h) Complete Network Training Motion Trajectory Loss

Figure 3.19: Training Error Plots for DNN-CPG architecture in two steps, experiment 10

CHAPTER 4

REWARD FORMULATION

4.1 Introduction

Reinforcement learning is concerned with how intelligent agents ought to take actions in an environment to maximise the notion of cumulative reward. Thus, to allow the DNN-CPG to learn sophisticated behaviour, a reward was formulated for each behaviour to be learnt. For instance, a Zero Moment Point based reward was formulated to maintain a stable gait despite challenging terrain. The reward formulated must quantify an abstract quantify like gait, locomotion efficiency and obstacle negotiation. Thus, the reward must be grounded in the physics of quadruped locomotion.

4.2 Stability Reward

The Stability reward quantifies the goodness of gait. Researchers have proposed a variety of static- and dynamic-stability-analysis methods to evaluate the stability of legged robots. The static-stability criteria and margins are only applicable for analysing low-speed static gaits. For instance, a stability criterion named wide stability margin (WSM) was proposed in [21]. WSM is the shortest distance from the projection of the centre of mass (CoM) of the body within the support polygon formed by the projection of the current support and swing feet on the horizontal plane to the boundaries of the support polygon.

On the other hand, the most common dynamic-stability criteria are based on the zero moment point (ZMP). ZMP refers to a point on the ground where the net torque in the direction parallel to the ground caused by the gravity and inertial forces of each part of the robot is zero. ZMP shows the tendency of the robot to tumble. Researchers believe that the robot is stable if ZMP is located inside the support polygon, connected to the support feet. The stability criteria used for reward formulation is based on a modified definition of ZMP proposed in [7].

4.2.1 Zero Moment Point

Let the CoM C of the quadruped be at $P_C^i = [x_C^i, y_C^i, z_C^i]^\top$, the weight be $G^i = [G, 0, 0]^\top$, the inertial force at CoM be $F_I^i = [F_{Ix}^i, F_{Iy}^i, F_{Iz}^i]$ and the inertial moment at CoM be $N_I^i = [N_{Ix}^i, N_{Iy}^i, N_{Iz}^i]$. If the position of the ZMP is $P_{ZMP}^i = [x_{ZMP}^i, y_{ZMP}^i, z_{ZMP}^i]^\top$, then according to the definition of ZMP,

$$(P_C^i - P_{ZMP}^i) \times F_I^i + (P_C^i - P_{ZMP}^i) \times G^i + N_I^i = 0 \tag{4.1}$$

Equation 4.1 can be expanded as follows according to the coordinate system in Figure 4.1.

$$-(z_C^i - z_{ZMP}^i)F_{Iy}^i + (y_C^i - y_{ZMP}^i)F_{Iz}^i + N_{Ix}^i = 0 \tag{4.2}$$

$$(z_C^i - z_{ZMP}^i)(F_{Ix}^i + G_x^i) - (x_C^i - x_{ZMP}^i)F_{Iz}^i + N_{Iy}^i = 0 \tag{4.3}$$

$$-(y_C^i - y_{ZMP}^i)(F_{Ix}^i + G_x^i) + (x_C^i - x_{ZMP}^i)F_{Iz}^i + N_{Iz}^i = 0 \tag{4.4}$$

Only two of the three equations are linearly independent [22], and thus, they can not be used for calculating the position of the ZMP.

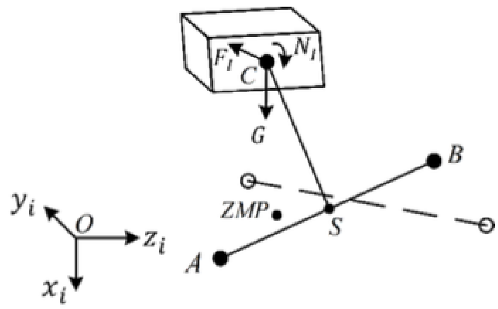


Figure 4.1: Force-schematic diagram of the robot body [7]

4.2.2 Support Plane and Coordinate System

Based on the assumptions of traditional ZMP, it may be stipulated that the ZMP lies on the current support plane. The support plane is a possibly non horizontal plane containing the support area of a robot. A coordinate system, called the support coordinate system may be defined for convenience. The origin of this coordinate system lies at the mid point of the support line $\overline{AB}$. The z-axis of the coordinate system is from the rear foot A to the front foot B. The x-axis is perpendicular to $\overline{AB}$ and points downwards. Figure 4.2 depicts the settings of the support coordinate system.

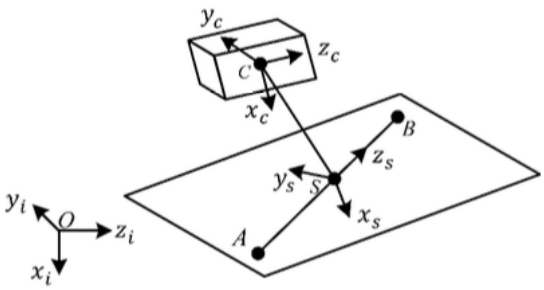


Figure 4.2: Settings of the support coordinate system [7]

The position of the ZMP is $P^s_{ZMP} = [x^s_{ZMP}, y^s_{ZMP}, z^s_{ZMP}]$ in the support coordinate system such that,

$$x^s_{ZMP} = 0 \tag{4.5}$$

$$y^s_{ZMP} = y^s_C - \frac{x^s_C(F^s_{Iy} + G^s_y) + N^s_{Iz}}{F^s_{Ix} + G^s_x} \tag{4.6}$$

$$z^s_{ZMP} = z^s_C - \frac{x^s_C(F^s_{Iz} + G^s_z) + N^s_{Iy}}{F^s_{Ix} + G^s_x} \tag{4.7}$$

Since it is difficult to ascertain the support plane for dynamic gaits, a virtual support plane is defined. The positions of the current support line, $\overline{AB}$, previous support line, $\overline{A_F B_F}$ and next support line, $\overline{A_L B_L}$ are comprehensively considered for computing the virtual support plane. The previous support line is stored as the robot moves. The next support line is approximated using forward kinematics and prediction from the DNN-CPG. The computation of the next support line is discussed in detail in later sections.

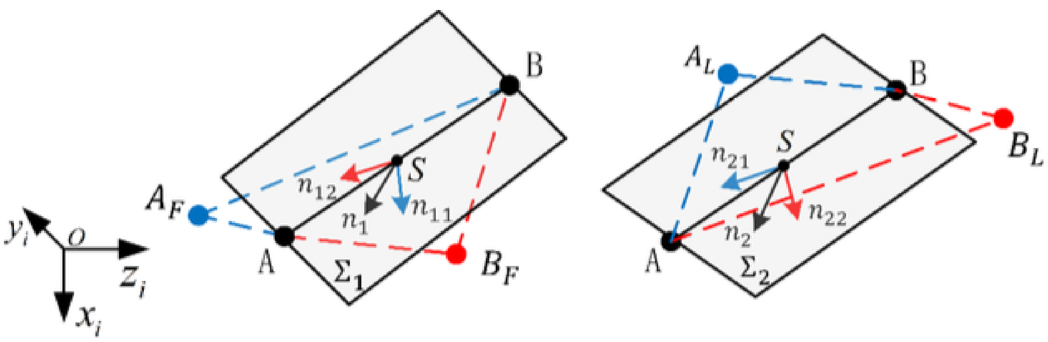


Figure 4.3: Determination of the Virtual Support Plane [7]

The transformation from the inertial coordinate system to the support coordinate system is calculated using the following equations,

$$n_{11} = \frac{\vec{AB} \times \vec{AA}_F}{\|\vec{AB} \times \vec{AA}_F\|} \tag{4.8}$$

$$n_{12} = \frac{\vec{AB} \times \vec{BB}_F}{\|\vec{AB} \times \vec{BB}_F\|} \tag{4.9}$$

$$n_{11} = n_{12} \Leftrightarrow \vec{AB} \times \vec{AA}_F = 0 \tag{4.10}$$

$$n_{11} = -n_{11} \Leftrightarrow n_{11x_i} \leq 0 \tag{4.11}$$

$$n_1 = \frac{n_{11} + n_{12}}{\|n_{11} + n_{12}\|} \tag{4.12}$$

$$n_{21} = \frac{\vec{AB} \times \vec{AA}_L}{\|\vec{AB} \times \vec{AA}_L\|} \tag{4.13}$$

$$n_{22} = \frac{\vec{AB} \times \vec{BB}_L}{\|\vec{AB} \times \vec{BB}_L\|} \tag{4.14}$$

$$n_2 = \frac{n_{21} + n_{22}}{\|n_{21} + n_{22}\|} \tag{4.15}$$

$$\mu = -\frac{t}{T_b} + 1 \tag{4.16}$$

where T_b is the duration between two adjacent steps and t is the time gap between the current running time and the time point when the previous support line disappeared. Using the above results, the axes of the support plane are defined as follows,

$$x_s^i = \frac{\mu n_1 + (1-\mu) n_2}{\|\mu n_1 + (1-\mu) n_2\|} \tag{4.17}$$

$$z_s^i = \frac{\vec{AB}}{\|\vec{AB}\|} \tag{4.18}$$

$$y_s^i = z_s^i \times x_s^i \tag{4.19}$$

4.2.3 Modified ZMP

The Modified ZMP definition takes into account the current velocity of the robot. The current velocity is an important factor in determining the stability of motion as stability may be attained in the next few time steps. This behaviour is similar to anticipation that animals show during motion. To imbue the DNN-CPG with such an anticipatory capabilites, the modified ZMP definition was used. The following equations define the modified ZMP in the support plane,

$$x^s_{ZMP_o} = 0 \tag{4.20}$$

$$y^s_{ZMP_o} = y^s_{ZMP} + \eta(v^s_{yr} - v^s_{yd}) \tag{4.21}$$

$$z^s_{ZMP_o} = z^s_{ZMP} + \eta(v^s_{zr} - v^s_{zd}) \tag{4.22}$$

$$\eta = 0.005\frac{L+W}{\|v_d\|} \tag{4.23}$$

where v_d is the desired velocity in the inertial coordinate system, v^s_d is the desired velocity in the support coordinate system, v^r_r is the real velocity in the support coordinate system and L is the length of the base of the quadruped and W is the width of the base of the quadruped. Since both the inertial coordinate system and the support coordinate system are known, the computed quantities can be easily transformed between the coordinate systems.

4.2.4 Reward

The reward to be used is a weighter sum of the following three quantities,

1. Distances between ZMP_o and the boundaries of the virtual-support quadrilateral in the support plane, d_{edge}
2. Angle between the vector pointing from CoM to ZMP_o and the normal vector of the force is perpendicular to the , $\sin\tau$
3. Distance between ZMP_o and the support line, d_{spt}

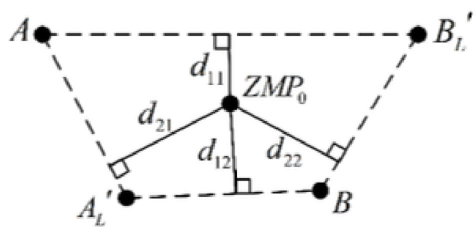


Figure 4.4: Distances between ZMP_o and each boundary of the virtual-support quadrilateral [7]

The distances between ZMP_o and the boundaries of the virtual-support quadrilateral in the support plane is used to compute the integrated minimum distance between

ZMP_o and the boundaries, d_{edge} defined by the following equations,

$$d_{edge} = min\{d_{11}, d_{12}\} + min\{d_{21}, d_{22}\} \tag{4.24}$$

Refer to Figure 4.4 for definition of d_{11}, d_{12}, d_{21} and d_{22}.

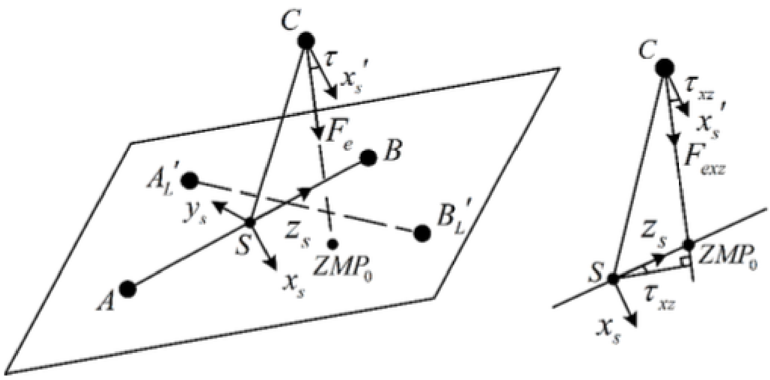


Figure 4.5: Angle between the vector pointing from CoM to ZMP_o and x_s^i [7]

The following equations define $\sin\tau$,

$$F_e = P^i_{ZMP_o} - P^i_C \tag{4.25}$$

$$\cos\tau = \frac{F_e \cdot x_s^i}{\|F_e\|\|x_s^i\|} \tag{4.26}$$

$$|\sin\tau| = \sqrt{1 - \cos\tau^2} \tag{4.27}$$

Refer to Figure 4.5 for details of the vectors in the above equations.

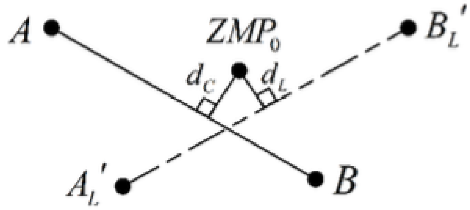


Figure 4.6: Distances between ZMP_o the support lines (d_C and d_L) [7]

The following equations define d_{spt} according to the distances defined in Figure 4.6,

$$d_{spt} = \omega_C d_C + \omega_L d_L \tag{4.28}$$

$$\omega_C = \begin{cases} 1, & 0 \leq t < 0.25T_b \\ -\frac{2}{T_b} + 1.5, & 0.25T_b \leq t < 0.75T_b \\ 0, & 0.75T_b \leq t < T_b \end{cases} \tag{4.29}$$

$$\omega_L = \begin{cases} 0, & 0 \leq t < 0.25T_b \\ \frac{2}{T_b} - 1.5, & 0.25T_b \leq t < 0.75T_b \\ 1, & 0.75T_b \leq t < T_b \end{cases} \tag{4.30}$$

The Stability Reward is a weighted sum of the three quantities defined by the equations above. The following equations define the Stability reward, $M_{stability}$,

$$M_{stability} = \lambda_1 d_{edge} - \lambda_2 |\sin\tau| - \lambda_3 d_{spt} \tag{4.31}$$

$$\lambda_1 = 1 \tag{4.32}$$

$$\lambda_2 = \frac{1}{8}(L + W) \tag{4.33}$$

$$\lambda_3 = \frac{9(L + W)}{40W} \tag{4.34}$$

4.3 Summary

Different Reward formulations serve to introduce different behaviours into the DNN-CPG. The Stability reward is used to fine-tune the walking behaviour learned through supervised learning. It serves to reduce the tendency of the robot to topple in challenging terrains. More sophisticated behaviour like efficient gait transition and obstacle negotiation can also be learned by formulating appropriate reward functions and incrementally training the DNN-CPG. The behaviour that will be tackled in this project are as follows,

1. Efficient Gait
2. Gait Transition
3. Obstacle Negotiation
4. Object Search

Appropriate reward function will be formulated for each behaviour and used for RL.

CHAPTER 5

REINFORCEMENT LEARNING

5.1 Introduction

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment to maximise the notion of cumulative reward. RL utilises the experience of the agent within the environment for learning. As learning progresses, the agent builds up knowledge of dealing with the challenges faced in the environment. An incentive or a reward drives this building up of knowledge. For animals, the reward is survival and the resulting release of chemicals that cause positive reinforcement of successful behaviour. In the case of the DNN-CPG, the reward consists of the rewards formulated in Chapter 4. Deep Deterministic Policy Gradient (DDPG) is used to perform RL with DNN-CPG.

5.2 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is a model-free off-policy algorithm for learning continuous actions. The DDPG combines the actor-critic approach with insights from the recent success of Deep Q Network. It is not possible to apply Q-learning to continuous action spaces because in continuous spaces finding the greedy policy requires optimisation of actions at every timestep. This optimisation is too slow to be practical with large, unconstrained function approximators and nontrivial action spaces. Instead, an actor-critic approach is used based on the Deterministic Policy Gradient algorithm [23]. Figure 5.1 depicts a simplified architecture of the DDPG algorithm. The actor-network produces the action ($\mu_\theta(s)$) given the state of the robot from the environment. The produced action and the state are then given as input to the critic network, predicting the Q-scores for all the produced actions. The predicted Q-scores are then used to compute the gradient for parameter update of both the Actor-network and Critic network parameters.

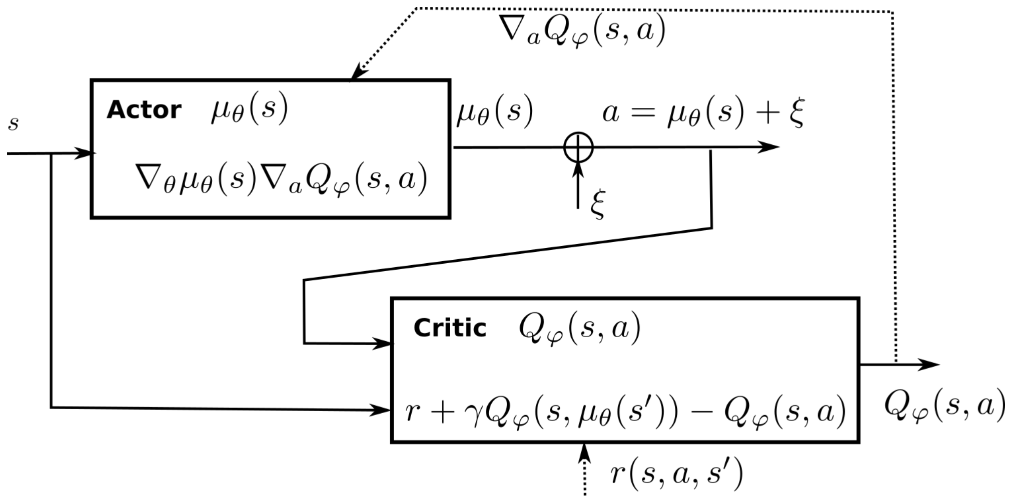


Figure 5.1: Architecture of the DDPG Algorithm [8]

5.2.1 DNN-CPG as Actor

The DNN-CPG serves as the actor of the DDPG, and the joint motion trajectories produced constitute the action space. Currently, the DNN-CPG consists of the motion state encoding network, the robot state encoding network, the CPG and the complex MLP. The training's current objective is to teach the DNN-CPG to produce motion trajectories that ensure stable gait despite challenging terrain. Figure 5.2 depicts the snapshots of DNN-CPG at a time step t predicting the joint activation A_t for that time step. This training stage is the first step of RL before all proposed in Section 4.3 behaviour can be learnt. The DNN-CPG takes the motion state, the robot state and the previous oscillator output as input.

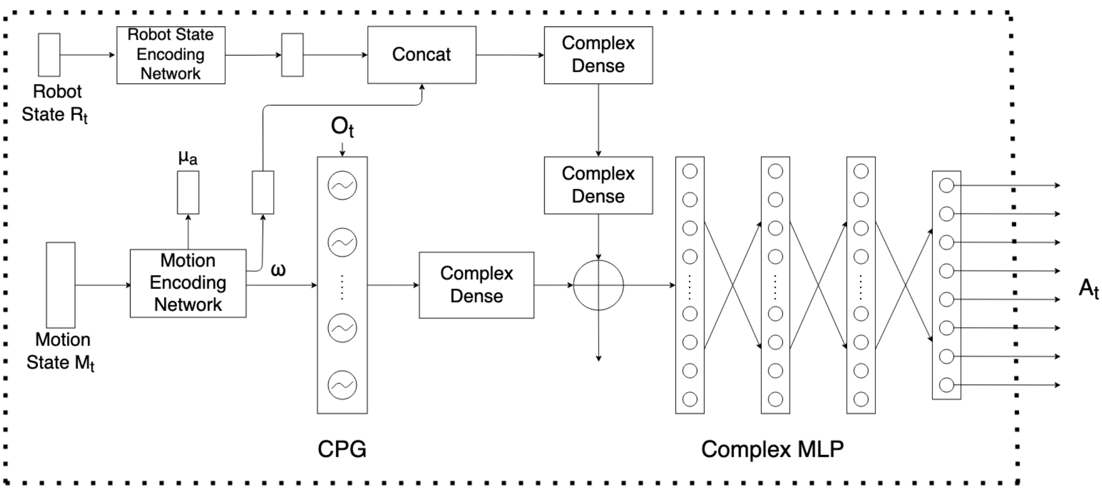


Figure 5.2: Snapshot of DNN-CPG at a time step t

5.2.2 Actor RNN

The DNN-CPG in Figure 5.2 can only produce the joint activations for a single time step. However, motion trajectories for the future are required as well for the calculation of the stability reward. The stability reward requires the knowledge of the next support line, given the current state of motion. To enable the network to produce future joint activations, the DNN-CPG is modified into a Recurrent Neural Network (RNN) as shown in Figure 5.3. The input of the RNN is the same as the DNN-CPG in Figure 5.2; that is, it takes as input the current motion state, the current robot state and the previous oscillator output. However, the output of the RNN is the joint activations for a predefined number of time steps and the first output from the oscillator. Such a network architecture allows for a sliding window-like scenario, wherein the first output of the RNN is used to drive the robot and produce the motion trajectories, whereas the remaining outputs are used to calculate the stability reward.

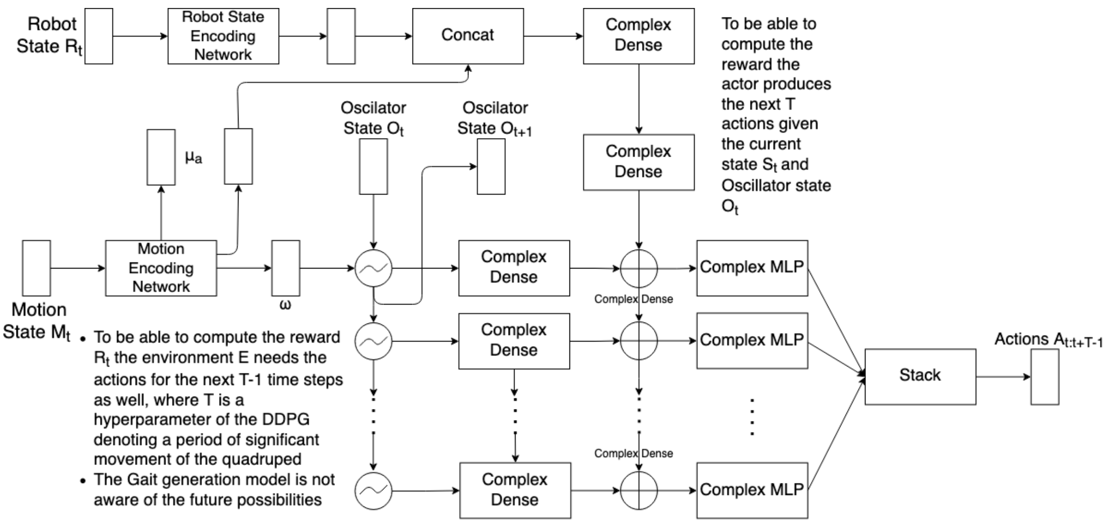


Figure 5.3: Actor RNN for production of future Joint Activations

5.2.3 Environment

In RL, the environment is the task or simulation that the agent interacts with and learns from. The agent produces an action that is input to the environment, which provides the agent with a state and computes the reward for driving the learning process. The MePed Quadruped Platform and its simplified quadruped description constitute the simulation that forms the environment of the RL problem at hand. The environment is also responsible for storing the state and action history for reward computation. For instance, the stability reward requires the knowledge of the previous support line of the quadruped. The environment stores this information. Figure 5.4 depicts the input-output pipeline of the environment developed for the project.

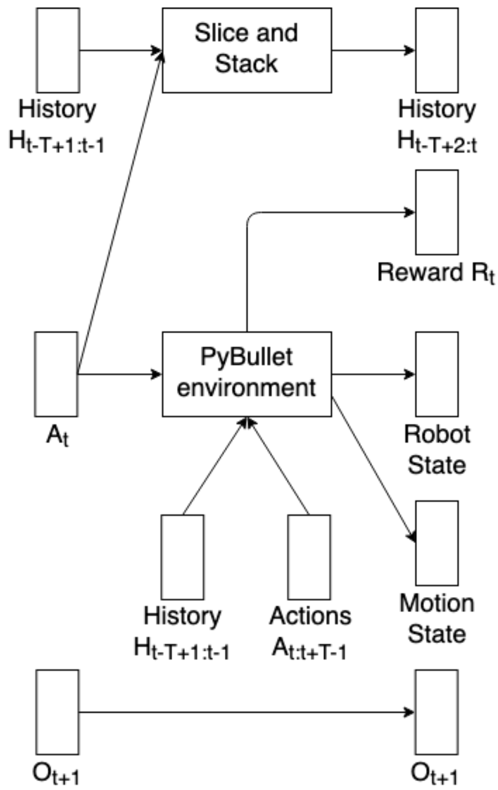
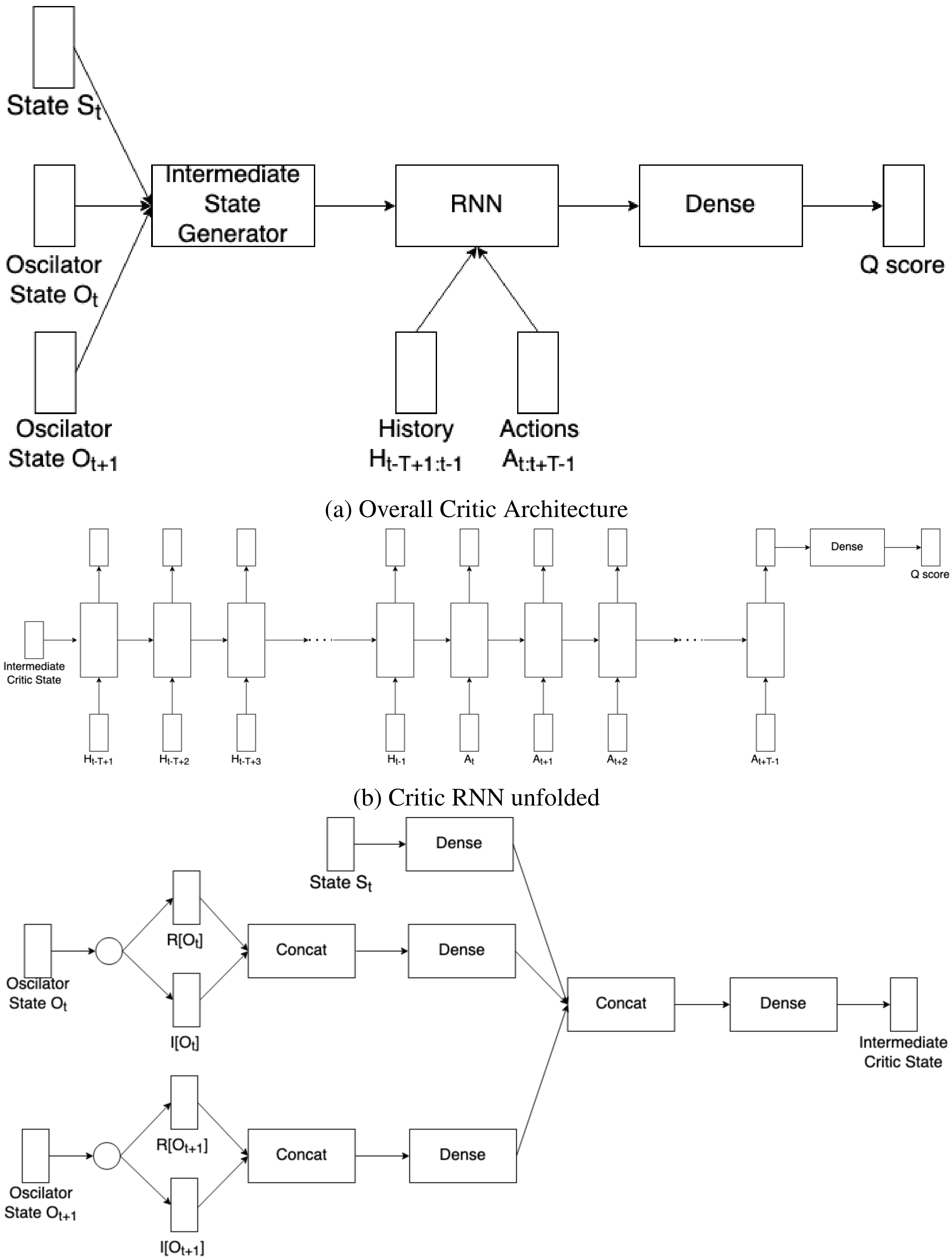


Figure 5.4: Input-Output pipeline of the environment

5.2.4 Critic RNN

The Critic drives the learning in DDPG. The Q-scores predicted by the critic network are used to compute the gradients for parameter updates of the Actor-network as well. Similar to the actor-network, the critic network is also defined as an RNN. This was done so that the Q-scores predicted to take into account the past and future joint activations. Such a formulation was necessary as the reward computation considers the robot's complete motion trajectory, and for efficient learning, it is hypothesised that an RNN would perform better. Figure 5.5 depict the Critic RNN architecture and its components. The Critic RNN consists of two major components,

1. The Intermediate State Generator that takes input the motion state, the robot state and the oscillator output and generates a real-valued state vector for the RNN



(a) Overall Critic Architecture

(b) Critic RNN unfolded

(c) Intermediate State Generator

Figure 5.5: Critic RNN Architecture and Components

2. The Critic RNN that takes as input the actions predicted by the Actor and the History of actions and outputs Q-Scores

5.3 Summary

The Actor and Critic Networks at the current stage are meant for fine-tuning the gait learnt through supervised learning. Training of the DDPG is performed on an AWS EC2 instance, utilising the NVIDIA T4 GPUs for acceleration. Once the current training step is completed, the next step would be to teach the quadruped more sophisticated behaviour of wandering and turning in an austere environment. By following an incremental approach to training, we ensure that each behaviour is learnt independently of other behaviours. Such a process also gives insight into the development of the brain's sensorimotor processing centres with experience as each module added to the DNN-CPG may be mapped to a specific functional unit in the brain.

REFERENCES

[1] Microsonic, “Ultrasonic technology,” https://www.microsonic.de/en/support/ultrasonic-technology/principle.htm, 2021, [Online; accessed 12-March-2021].

[2] N. Khomariah and S. Huda, “Walking Pattern for Quadruped as Observer Robot,” *Journal of Physics: Conference Series*, vol. 1201, p. 012010, may 2019. [Online]. Available: https://doi.org/10.1088/1742-6596/1201/1/012010

[3] L. Righetti and A. J. Ijspeert, “Pattern generators with sensory feedback for the control of quadruped locomotion,” *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 819–824, 2008. [Online]. Available: http://infoscience.epfl.ch/record/130740

[4] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, “From swimming to walking with a salamander robot driven by a spinal cord model,” *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007. [Online]. Available: https://science.sciencemag.org/content/315/5817/1416

[5] S. Auddy, S. Magg, and S. Wermter, “Hierarchical control for bipedal locomotion using central pattern generators and neural networks,” in *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2019, pp. 13–18.

[6] R. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[7] Y. Jia, X. Luo, B. Han, G. Liang, J. Zhao, and Y. Zhao, “Stability criterion for dynamic gaits of quadruped robot,” *Applied Sciences*, vol. 8, no. 12, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/12/2381

[8] Julien Vitay, “Deep reinforcement learning,” https://julien-vitay.net/deeprl/DPG.html, [Online; accessed 21-March-2021].

[9] S. Hirose and K. Kato, “Study on quadruped walking robot in Tokyo Institute of Technology-past, present and future,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, September 2000, pp. 414–419 vol.1.

[10] H. Kimura, Y. Fukuoka, and H. Katabuti, “Mechanical design of a quadruped ” tekken 3 4 ” and navigation system using laser range sensor,” 2005.

[11] D. Hyun, S. Seok, J. Lee, and S. Kim, “High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah,” *The International Journal of Robotics Research*, vol. 11, no. 2, pp. 1417–1445, August 2014.

[12] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “Bigdog, the rough-terrain quadruped robot,” in *Proceedings of the 17th IFAC World Congress*, 2008.

[13] X. Meng, S. Wang, Z. CAO, and Z. Leijie, "A review of quadruped robots and environment perception," in *2016 35th Chinese Control Conference (CCC)*, July 2016, pp. 6350–6356.

[14] J. H. BarronS-Zambrano and C. Torres-Huitzil, "CPG Implementations for Robot Locomotion: Analysis and Design," in *Robotic Systems*, A. Dutta, Ed. Rijeka: IntechOpen, 2012, ch. 9. [Online]. Available: https://doi.org/10.5772/25827

[15] A. Sproewitz, R. Moeckel, J. Maye, and A. J. Ijspeert, "Learning to move in modular robots using central pattern generators and online optimization," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 423–443, 2008. [Online]. Available: https://doi.org/10.1177/0278364907088401

[16] C. P. Santos and V. Matos, "CPG modulation for navigation and omnidirectional quadruped locomotion," *Robotics and Autonomous Systems*, vol. 60, no. 6, pp. 912–927, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889012000164

[17] R. J. Szadkowski, P. Cizek, and J. Faigl, "Learning central pattern generator network with back-propagation algorithm," in *Proceedings of the 18th Conference Information Technologies - Applications and Theory (ITAT 2018), Hotel Plejsy, Slovakia, September 21-25, 2018*, ser. CEUR Workshop Proceedings, S. Krajci, Ed., vol. 2203. CEUR-WS.org, 2018, pp. 116–123. [Online]. Available: http://ceur-ws.org/Vol-2203/116.pdf

[18] H. Lee, Y. Shen, C. han Yu, G. Singh, and A. Y. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *in In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 2006.

[19] H. Liu, W. Jia, and L. Bi, "Hopf oscillator based adaptive locomotion control for a bionic quadruped robot," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, September 2017, pp. 949–954.

[20] L. Zhou, W. Liu, H. Qian, and Y. Xu, "Gait design and comparison study of a quadruped robot," 07 2017.

[21] H. Kimura, Y. Fukuoka, and A. Cohen, "Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, 2007. [Online]. Available: https://doi.org/10.1177/0278364907078089

[22] M. Khorram and S. A. A. Moosavian, "A 3d stable trot of a quadruped robot over uneven terrains," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 231, no. 3, pp. 555–573, 2017. [Online]. Available: https://doi.org/10.1177/0954406215617492

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.