DESIGNING REINFORCEMENT LEARNING FRAMEWORK FOR A FINITE STATE MACHINE

A thesis submitted to the
University of Petroleum and Energy Studies

For the Award of
Doctor of Philosophy
in
Computer Science

By
Rashmi Sharma
(SAP ID 500024015)

June 2020

Internal Supervisor
Dr. Inder Singh
Assistant Professor (S.G.)
School of Computer Science
University of Petroleum & Energy Sciences

External Supervisor
Dr. Ashutosh Pasricha
OFS Director
Schlumberger- India

UNIVERSITY WITH A PURPOSE

University of Petroleum & Energy Studies
School of Computer Science
University of Petroleum & Energy Studies
Dehradun-248007: Uttarakhand

January 2021

DECLARATION

I declare that the thesis entitled "Designing Reinforcement Learning Framework for a Finite State Machine", has been prepared by me under the guidance of Dr. Inder Singh, Assistant Professor (S.G.) of Computer Science, University of Petroleum & Energy Studies, and Dr. Ashutosh Pasricha, OFS Director, Schlumberger - India, Member of Academic Council, University of Petroleum & Energy Studies. No part of this thesis has formed the basis for the award of any degree or fellowship previously.

Rashmi

Rashmi Sharma

DATE: 21 Jan 2021

UPES
UNIVERSITY WITH A PURPOSE

Great Place To Work® Certified MAR 2020–FEB 2021 INDIA

CERTIFICATE

I certify that Rashmi Sharma has prepared her thesis entitled **"Designing Reinforcement Learning Framework for a Finite State Machine"**, for the award of the Ph.D. degree of the University of Petroleum & Energy Studies, under my guidance. She has carried out the work at the Department of Computer Science Engineering, University of Petroleum & Energy Studies.

Internal Supervisor

[illegible]

Dr. Inder Singh

Assistant Professor (S.G.)

School of Computer Science

University of Petroleum & Energy Sciences

Date: 21-01-2021

CERTIFICATE

I certify that Rashmi Sharma has prepared her thesis entitled **“Designing Reinforcement Learning Framework for a Finite State Machine”**, for the award of the Ph.D. degree of the University of Petroleum & Energy Studies, under my guidance. She has carried out the work at the Department of Computer Science Engineering, University of Petroleum & Energy Studies.

External Supervisor

[illegible]

Dr. Ashutosh Pasricha

OFS Director,

Schlumberger - India

Date:

ABSTRACT

The scope of this research was to design a framework with RL for autonomous mobile/ bipedal robots. The result was designing, programming, and validation of RL based algorithms for navigation of the Bipedal Walking Robot. The improvements proposed include 1. Incorporation of Forgetting Mechanism in Traditional Q-learning Algorithm 2. Feature-based Object Identification by the RL agents in the dynamic environment. 3. Hierarchical Training of the RL agent. RL agent is a Bipedal in this case. This research work examines improvements in traditional Q-learning algorithms to successfully interact with a dynamic and uncertain environment. Simulations were carried out for each proposal. Incorporating the Forgetting mechanism resulted in a considerable improvement in the learning time of RL agents (Hip joint, Knee joint, Ankle joint) in a dynamic environment. The Feature-based Object Identification Algorithm reduced considerably in the number of state values that are required to be maintained. This facilitates the use of multiple agent systems (MAS) in large environments with dynamic conditions. The Hierarchical implementation of the algorithm help in sharing and transferring the knowledge from one RL agent to another RL agent. Also useful for obstacle avoidance and identifying dangerous objects while navigating. The communication and data sharing between MAS are online as well as offline to that the bipedal walk's without tipping and with stability.

It uses different modules that consist of simple controllers with RL forgetting the Q-learning algorithm. The feature-based object identification system would help to identify objects and the bipedal controller would be able to take appropriate actions. The present work deals with vision-based navigation (VBN) of bipedal. The bipedal identifies the object by using an updated SURF algorithm.

The reinforcement control algorithms for the Bipedal robot had been applied for self-learning and for taking self-decision. Bipedal is sensing the present state and switching to the next subsequent stable state and finally reaching the desired goal state. Simulation is carried out on the MATLAB platform and SimSpace Multibody dynamics toolbox to verify proposed algorithms. The optimal policy is achieved in reaching the goal/target state and are stored in the lookup tables for future use. After the learning of each of the agents of MAS is completed, the execution phase starts. The data from the lookup table is visited for further decision making. The data is stored in a lookup table which helps in reducing the learning time of the agents. As the number of strides increases the size of the lookup table increases, the agent gets more options for exploration of a new stable state. Then after a few runs, it starts exploiting the explored data in terms of the next state and the optimal policy previously achieved. These lookup tables are useful if the scenario in the dynamic environment does not change. This helps in reducing the execution time of the multiple agents. When the scenario changes the agents will learn from scratch. Several experiments are carried out on MATLAB to verify the analytical and simulation result. The results verify that the execution time reduces to a considerable amount. There is overhead attached when the size of the lookup table increases beyond the limit its search time increases. This results in approximately the same time for the learning and execution phase of the bipedal gait.

ACKNOWLEDGMENTS

I, first and foremost, would like to express my deepest gratitude to Dr. Manish Prateek (Mentor), Dr. Inder Singh (Internal Supervisor), Dr. Ashutosh Pasricha (External Supervisor) for encouraging me to take this research problem on Designing of Reinforcement Learning Framework for a Finite State Machine. My mentor and supervisors were always a source of inspiration and source of motivation to me throughout my research work. Without them, this work would never have been completed.

I am thankful to Chancellor Dr. S. J. Chopra, Vice-Chancellor Dr. Sunil Rai, Dr. Kamal Bansal, Dr. Jitendar Pandey. I would like to mention a special thanks to team R&D UPES for providing constant guidance and support.

I am extremely grateful to my doctoral member Dr. Deepak Bharadwaj for his constant encouragement and fruitful discussion during the work. He helped me while implementing the software and theoretical framework of this research. Also, I am thankful to each faculty member of Computer Science Engineering for their support and cooperation.

I would like to thanks Ms. Rakhi Ruhal, Mr. Sony Sandeep Farmer, and other staff members for their help and numerous occasions and support.

Finally, I must express my very profound gratitude to my parents Prof. Someshwar Trivedi, Ms. Jyotsna Trivedi, my in-laws late Shri Ved Ram Sharma, Ms. Shiksha Sharma, my sisters Ms. Shikha Trivedi and Ms. Shardha Trivedi, and other family members for their support and numerous blessings.

Most importantly, I thank my husband Mr. Yogendra Kumar Sharma, and my daughter Ms. Pranjali Sharma, my son Master Mantavya Sharma for always being there with me and being a persistent source of encouragement.

I am thankful to God who converted the positive and negative energy present to me into some creative output.

Rashmi Sharma

UPES Dehradun

June 2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI :	Artificial Intelligent
ASIMO:	Advanced Step in Innovative Mobility
DSP:	Double Support Phase
DOF:	Degree of Freedom
iCub:	Cub Standing for Cognitive Universal Body
IP:	Inverted Pendulum
IPM:	Inverted Pendulum Model
MDP :	Markov Decision Process
PETMAN:	Protection Ensemble Test Mannequin
RL:	Reinforcement Learning
SSP:	Single Support Phase
SURF :	Speeded Up Robust Features
VBN :	Vision Based Navigation
ZMP:	Zero Moment Point

LIST OF SYMBOLS

$I_{\Sigma}(X)$: Sum of areas of location $X=(x, y)^T$

$H(A,\sigma)$: Hessian Matrix in A at scale σ

$Lxx(A,\sigma)$: Convolution of 2nd order Gaussian derivative $(g(\sigma))^{\cdot\cdot}$ at point A of an image I

$Lyy(A,\sigma)$: Convolution of 2nd order Gaussian derivative $(g(\sigma))^{\cdot\cdot}$ at point A of an image I

D_{xx}: Approximation of 2nd order Gaussian partial derivative in the X-direction

D_{yy}: Approximation of 2nd order Gaussian partial derivative in the Y-direction

d_x: Haar Wavelet responses in the x-axis

d_y: Haar Wavelet responses in the y-axis

D_E/D_M: Euclidean or Mahalanobis distance

f_n: Natural frequency of the 2-D inverted pendulum

l: Distance between ground and center of mass (Pendulum length)

g: Gravitational acceleration

T: Torque measured at the hip/ knee/ ankle joint

m: Point mass

F_z: Ground reaction force

Y_{mc}: Lateral displacement of the mass center

Y_{zmp}: Lateral ZMP

X_{mc}: Forward displacement of the mass center

X_{zmp}: Forward ZMP

θ: Actual joint angle due to compliance (hip/ knee/ ankle)

u: Reference joint angle

K: Stiffness of the leg

k_d: Damping control gain

u_c: Compensated joint angle

Y_{pelvis}: Lateral displacement of the pelvis

α: Learning rate

λ: Discount Factor

ε: Exploration probability (Exploration (ε)/Exploitation(1- ε))

ε-decay: Forgetting factor (epsilon decay)

S_t: Current/ Present State

S_p: Processed state

a_t : Current/ Present action

S_{t+1}: Next subsequent stable state

r_{t+1}: Immediate reward

r_t: Delayed reward

CHAPTER 1 INTRODUCTION

Humans have always been fascinated by creating creatures like them, which resulted in the designing and development of Humanoid/ Bipedal Robots. The bipedal developed should be able to act, interact like a human being, and should be intelligent enough to reason. Bipedal have a human body which through morphological calculation ought to naturally adjust and make up for movement and dynamic conduct.

Bipedal along with the locomotion should also integrate performing tasks associated with manipulation, perception, interaction, adaptation, and self-learning. This leads to a connection with legal, social and ethical domains along with science and engineering disciplines. Bipedal are cross-disciplinary including propelled velocity and control, biomechanics, computerized reasoning, machine vision, recognition, learning, and subjective improvement alongside the social examinations.

Bipedal are best suited till now for the predefined tasks like in automotive fields, as a companion in medical surgery, cleaning and mopping the floor, mowing the garden, and so on. Bipedal cannot perform tasks in an unstructured and dynamic environment. Bipedal learns from mistakes as a human being and adapts to the dynamic and uncertain environment with the algorithms developed by humans. Bipedal should learn the dynamics of the environment similar to a child, learn to walk/ crawl in a dynamic environment that is changing for every go of the walk. As the child learns from mistakes and failures, walking an unforgettable task of life. Similarly, with machine vision, artificial intelligence (AI), cognitive learning algorithms, bipedal can adapt to the uncertainty of the environment and can accomplish their desired goals. Implementation of bipedal to the community is a social, economic, and legal issue.

The aim is to ease human efforts, save human life in a hazardous environment. The future of the bipedal robots is like an emotional, physical companion of humans, which can help humans in household chores, in a hazardous environment, as a companion and friend at the workplace.

1.1 History of Humanoid/ Bipedal Robot

The word 'Humanoid Robot' describes creatures that resemble humans and can be used to do tedious and hazardous tasks.

In 250 BC, Liezi described automation as a self-operating machine that is designed to follow the predefined task automatically. In 50 AD, Alexandria, a Greek mathematician depicted a machine that consequently pours wine for the gathering visitors. Al Jazari, in 1206 made handwashing automata with programmed robot hirelings and clock having elephants and mahout. He additionally portrayed a band of humanoids that can be performed more than 50 facial and posture activities during their musical show.

Jacques de Vaucanson, in 1738 structured a woodwind player that resembles a shepherd and had the option to play twelve tunes on woodwind. He additionally created a tambourine that played woodwind and drum. In 1774, Pierre Jacquet Droz developed animated dolls which helped the firm in selling watches. Later his son Henri Louis created a figure of a boy like a Draughtsman who can remember 40 characters' messages.

Karel Capek, in 1921, coined the 'Robot' word which was derived from 'robota' meaning 'to work'. In 1927, Mashinenmeusch (machine-human) humanoid robot additionally called Parody/ Futura/ Robotrix humanoid showed up in the film. In the same year, the humanoid robot Herbert Televox was developed by Ron Wensley. The humanoid could lift the receiver to answer a call and correspondingly controlling the task with the help of a switch. This robot did not have any ability to speak. In 1928, Eric's electrical robot opened a presentation to the general public of model designers in London and visited the world.

In 1941-42, Isaac Asimo gave three laws of robotics that deals with the safety restrictions of the robot. He used these laws in his science fiction stories. The story was recompiled in 'I' robot movie in the year 2004.

In 1961, Unimate, the first carefully worked programmable non-humanoid robot was introduced in sequential construction systems and assembly lines of General Motors. It was utilized to lift hot bits of metal from bite the dust throwing (die-casting) machine.

In 1967-72, initiated in 1967 and completed in 1972. WEBOT 1, which is the world's intelligent humanoid robot. WEBOT 1 was the first android which can walk, communicate in Japanese with people, measure distances and direction of the objects, grips, and transports object with hands.

In 1970, Miomir proposed a hypothetical idea of Zero Moment Point(ZMP). In 1972, Miomir and his partner construct the first dynamic human exoskeleton.

In 1980, Marc Raibut set up MIT leg Lab, devoted to legged velocity and building dynamic legged robots. In 1983, 'Green man' was developed which had a torso, arm, and head, the vision system consists of a camera that was mounted on the helmet. In 1984, WABOT 2, a musician humanoid was created. It communicated with persons, can read normal musical notes. In 1986-1993, Honda developed seven biped robots E0-E6. In 1989, Hanny a full-scale anthropomorphic was developed by the US. It can crawl and had 42 DOF.

In 1990, the Bipedal mechanical structure with knees was developed by Tad McGeer, which was even able to walk on a sloppy surface. In 1993, Honda developed P1 to P3 with upper limbs. In 1995, Webian, a human-sized biped walking robot was developed. In 1996-98, Saika light-weight with 2 DOF in the neck, double 5 DOF upper arms, body, and the head was developed. In 2000, Honda created its 11th bipedal which was capable of jumping, running, climbing stairs.

After 2000, Humanoids were at boom and their development was at the full pace some of them are listed here.

In 2001 HOAP-1, in 2002 HRP-2, in 2003 HOAP-2, JOHNNIE, ACTROID, in 2004 Persia, KHR-1, in 2005 HOAP-3, WAKAMAN, in 2006 iCUB, MALIM, in 2008 JUSTIN, in 2010-11 ROBONAUT-2, ASIMO (with semi-autonomous capabilities), in 2012 COMAN (Compliant Humanoid Robot), in 2013 SCHAFT, POOPY, in 2014 MANAV, PAPER ROBOT, NADINE, in 2015 SOPHIA and so on.

1.2 Features of Humanoid/ Bipedal Robots

The features, which a bipedal possesses are:

1. Autonomous Maintenance
2. Autonomous learning
3. Avoiding destructive circumstances to an individual's property and itself.
4. Self-interacting with human being and environment

1.2.1 Manipulation tasks

The manipulating ability of the bipedal robotic mechanisms is required to enter human-centered environments such as in positioning and orienting end-effectors. Due to a huge number of DOF, humans can manipulate objects of many shapes, sizes, weights, and materials(Asfour et al., 2008). Due to more number of joints and links, the singularity posture of the bipedal robot reduces the dexterity of the humanoid robot(Ott et al., 2006). Due to a lack of learning ability and limited manipulation of the bipedal robot, it is unable to pick new shape objects in an unknown environment. Many tasks require certain complaint behavior to make deliberate physical contact with the environment which are implemented by modern robotic manipulators.

1.2.2 Vision system

Visual perception is essential for bipedal working in human environments. The vision system of the bipedal robot extracts information of the joint positions and manages its own body along with the obstacle avoidance(Okada et al., 2006). The vision system manipulates the tool's movement through its own body and handles the target object. A bipedal robot is required to manage tools and objects.

Vision systems extract information on the shape and size of the target from the external and dynamic environment. An overhead camera is employed to the bipedal robot for computing the desired goal location and screens to interpret the captured images(Michel et al., 2005). The image processing technology provides a direct and indirect estimation of the target in the dynamic environment. In principle, machine vision includes acquiring an image, process it using digital image processing and analysis techniques, and take decisions based on the extracted information(Kagami et al., 2003). The functions are involved in real-time 3D vision are a generation of the 3D depth map, 3D depth flow generation, and plane segmentation finder. Real-time depth map generation framework and target finder run on the bipedal body computer. The other complex vision as the face recognizer and plane finder runs on the network computers.

1.2.3 Sensing behavior

The sensing behavior of the bipedal robot is the mapping of sensory input to different joint motor actions. This behavior is the appropriateness of the bipedal robotic response to a given task and environment. The reactive behavior provides a bipedal robot to interact with dynamic and unknown conditions without planning. These behaviors of the bipedal robot deal with targets independently and coordinating different joints in the desired way. Based on the environment interaction several sensors are utilized in bipedal. Force, tactile sensors are elementary sensors for the bipedal robot when interacting with the environment(Song et al., 2015). Bipedal robots entirely depend on the fusion of multiple sensors to provide them with information about their surroundings. The sensory input to the bipedal robot helps to understand the environment and navigation.

1.2.4 Mobile platform

Bipedal robots are moving towards applications beyond structured the environment(Khatib, 1999). The current generation of the bipedal robot has a mobile platform. Due to the mobile platform, a bipedal robot is entering the everyday world that people inhabit. Wheel based humanoid robot is working in a static environment. The introduction of the legged mobile platform into the bipedal robot will assist the bipedal to perform the task in the unstructured and dynamic environment.

1.3 Application of Bipedal Robot

1.3.1 Home Management Services

The bipedal robot observes the house in the absence of people and can be controlled remotely by people with the help of a simple mobile terminal. Humanoid robots can perform household activities. They can check the apparatus condition inside the house(Sawasaki et al., 2004).

1.3.2 Healthcare

The verbal and gesture interaction of bipedal robots can serve the patient in the hospital environment(Dahl & Boulos, 2013). Bipedal robot assisting nursing staff in taking care of the patients and in providing support while physically efficiently handling patients.

1.3.3 For aging / old aged people

Older age people would like a bipedal robot for assisting with daily routine tasks. With recent advancements in the technology of the bipedal robot, it can prevent an old aged person from falling, giving medication alerts, and managing their location(Robinson et al., 2014). These tasks need to maintain independence and dignity. The old age people are not able to take care of themselves. It is also disrespectful for them to use a device that looks like a toy. The appearance of the bipedal robot was fascinated. Bipedal are usually appreciated as they are capable to have improved communication with old age people and healthcare professionals both.

1.3.4 Industrial Application

The collaboration of humans and the bipedal robot contributes to the sustainable growth of factories. A place in which bipedal robots and people can work together to achieve goals (Maurtua et al., 2017). Safety is the most critical aspect of the industry. A bipedal robot looks after the safety of workers inside the industry while the execution of the task. Some tasks are complex and dangerous to be performed by a human. They require engineering special tools. An effective bidirectional human-robot communication contributes to the growth and safety of industrial development.

1.3.5 Space Exploration

With the advancement in satellite technology, satellites are deployed into space for exploration and colonization of other planets (Tanaka et al., 2017). The aim of these efforts to establish the possibility of life on other planets. Such plans require the creation of living environments on the planet. Bipedal robots can work and assist in space to establish a living environment. The bipedal robot can interact with the unknown environment of space can interact with the astronaut.

The application of a bipedal robot is not limited to these areas only. It can be used in different fields of industrial and non-industrial applications.

1.4 Motivation

The child suffering from the autism spectrum disorder disease deficit in social interaction and communication with the real world. Similarly, the old aged cannot do household activities on their own, walking machines resembling humans, bipedal plays an important role in their life. Interaction and self-decision making bipedal recognize eye contact and behavior of the old aged person and the child. After recognizing these parameters, bipedal can execute tasks desired by them. Till today, partial human thinking behavior implemented in the bipedal robot.

Open situations require a robot to design and learn under novel conditions. This must be done in a way that guarantees the wellbeing of the system and general environmental condition, and permits model estimation and figuring out how to occur inside a possible measure of time. Bipedal not exclusively be utilized as a partner yet additionally can be implemented into repetitive, tedious, and risky circumstances, for example, salvage tasks or bomb arranging. For Bipedal it is possible to do assignments dangerous for the individual. The assignments are possibly any perilous natural environment, for example, fire fighting operation, explosives, and can likewise aid other increasingly intricate, complicated, and confounded assignments. Currently, the most basic issue for bipedal is how to walk consistently in questionable and ceaselessly evolving conditions which is dynamic. Numerous scientists have controlled the ZMP position for strolling steadiness(Lohmeier et al., 2009)(Lowrey et al., 2018)(Ly et al., 2004). Applying appropriate walking gaits to biped walking would make the robot walk more stably and walking posture would resemble human walking.
The motivation of the present research work is implementing human thinking in bipedal, to serve social services to society. The dynamics of each humanoid's environment is different and so they cannot be trained for the static environment. Bipedal should learn to walk on its own as the scenario of its path changes.

1.5 Research Contribution

- The major contribution of this thesis is the development of model-free based reinforcement learning control calculation for an autonomous self-decision bipedal robot.
- The other contribution is to train bipedal to walk stably in the dynamic and uncertain environment when the position of objects differs in the environment.

- Another contribution is to train the bipedal to identify the object and localization of the object in the dynamic environment. If the same scenario is used, the previously learned data is utilized.

1.6 Thesis Outline

- Chapter 2 describes the history of humanoid/ bipedal robot, current ages of humanoid/ bipedal robot, bipedal robot movement, a mechanical model of bipedal robot, control design of humanoid/ bipedal robot and development of model-free based reinforcement.
- Chapter 3 describes the biomechanics of the lower body of the bipedal robot. Biomechanics deals with the motion and the orientation of each of the joint positions. The design input parameters considered from the standard human measurement and decided the link length and the joint trajectory.
- Chapter 4 introduces the general framework which is intended to accomplish the desired sub-objectives. The sub-objectives are objects identification and localization, bipedal control mechanism along with Reinforcement Learning control mechanism, and then the hierarchical structuring of all RL agents.
- Chapter 5 introduces the mathematical modeling of the bipedal robot. The gait trajectory and smooth motion evaluated for the different conditions. The kinematic and dynamic model (online/ offline) was implemented in the MATLAB platform and validated with the Multibody Toolbox of MATLAB. It also includes the object identification model along with the localization of the identified object.
- Chapter 6 introduces the design of the object identification algorithm of the bipedal. This includes a control framework for the feature-based identification of the object in the reinforcement learning control mechanism as proposed in this research work.
- Chapter 7 introduces the design of the reinforcement-learning controller of the bipedal. This also includes the proposed forgetting

mechanism incorporated in the traditional Q-learning Algorithm, how to model, and simulate Multi-Agent System (MAS).

- Chapter 8 combines both the proposed algorithm to form a model of the proposed framework which includes both algorithms with setting up the parameters for the executing simulated system so that the output produced is in sync with the actual output on the Bipedal. The simulation implemented in MATLAB platform and interfacing is done with the proposed framework.
- Chapter 9 describes the constants taken into consideration for simulation. The output is stored for future use in graphical and lookup tables for each of the joints. This stored data is used further as knowledge when the bipedal reach a similar condition when the dynamic environment is the same. The storage incurs time which was compensated by a reduction in the execution time of the bipedal by utilizing previous knowledge. This includes result graphs in 25, 50, 75, 100, 150, 200 strides for comparative study.
- Chapter 10 concludes and summarizes the research work on the designing, modeling, and simulating along with conclusion for the proposed algorithms so that the bipedal walks with stability along with the suggestions for future work regarding consideration of the upper body of bipedal along with an alternate way to store the optimal policy so that less time is required for reading and storing that data.

CHAPTER 2 LITERATURE REVIEW

Designing and developing the bipedal robots result in achieving real-world work at a greater speed and accuracy. Research communities and companies have been doing continuous work on the locomotion of humanoids in different environments - houses, fire rescue operations, coal mines, etc. In robotic research, the navigation of the bipedal is a challenging and emerging field. In this navigation, the bipedal should not get damaged nor any human being. This has gained significant attention as the dynamics of the environment in each case are continuously changing.

This reveals the major problem in designing the bipedal is self-awareness of the dynamics of the uncertain environment by the bipedal. Bipedal robots are also known as service robots which are included as assistant/companion robots by humans. Bipedal are used as medical assistants and teaching aids. The anthropomorphic form of bipedal robots offers greater flexibility for operating them in a different dynamic environment.

2.1 Current Humanoids / Bipedal

2.1.1 Vyommitra (Jan 2020)

Indian Space Research Organization (ISRO) introduced Vyommitra, half humanoid, legless but can bend forward and sideways. Vyommitra has a female look. Vyommitra means space (Vyoma) and a friend (Mitra). She can switch panel operation, environment control, and life support systems (ECLSS), monitor module parameters, be a companion and converse with the astronauts, alert the astronauts, and can respond to the queries. She will be sent to space as a trial before Gaganyaan, projected in 2022.

She simulates human functions in space, can check whether the system is right. She would help to monitor how the human system will behave in ECLSS. The robot, powered by speech synthesis software and artificial intelligence. The robot is seated at the desk in uniform and sported a custom-made ISRO identify badge with her name.

2.1.2 Sophia (Feb 2016)

Sophia is a humanoid that was designed by Hanson Robotics which can show 60 facial expressions. She is the only robot who is a citizen of Saudi Arabia. According to Davis Hanson, manufacturer, Sophia uses AI, facial recognition, visual data processing, voice recognition (speech to text) technology, and speech synthesis ability. These systems help her to be more brainy by collecting information from time to time. The program intelligently investigates and selects information that permits it to enhance future reactions. Sophia does not just keep up a keen talk with an individual on any subject, dialogues as also accompanied by emotional charges that help conversation between two people more normal.

The concepts work behind Sophia firstly include estimation of awareness of psychological framework, which uses tonomi phi values while reading and conversing. Secondly, Sophia is a realistic human-robot that can reproduce human to human social involvement in an excellent point of interest of controlled reputability towards patient or customer. Thirdly, Sophia helps in portraying robot behavior. Sophia uses AI techniques including the following face, acknowledgment of emotions, and mechanical movements created by a deep neural network. Sophia's discourse is created by the decision tree however is incorporated with these outputs exceptionally.

2.1.3 Atlas

Atlas is a bipedal humanoid robot planned by Boston Dynamics, funded by US DARPA (Defense Organization). It is supplied with two vision systems - a laser range discoverer and stereo cameras both constrained by an off-board PC. Atlas has hands with fine engine capabilities and has appendages. It can explore on uneven turf and can climb utilizing arms and legs. It utilizes sensors in its body and legs to adjust and to evade hindrances, evaluate the turf, help with routing, manage objects even when continuously moving.

Table 2.1 Body Parameters of Atlas

Parameter	Value
DOF	28
Tall	175 cm
Weighs	82 kg
Characteristics	Can jump on packages, turn 180° while hopping and doing a backflip

Atlas utilizes 3D printed parts which give it qualities to weight ratio vital for jumps of somersaults. To execute control loops inside the period dictated by a robot, real-time threads of the JNI library were used. Lockless synchronization natives are utilized to impart between the different threads, avoids garbage collection. The control arrangement of the robot is implemented on the robot operating system (ROS) packages(Maniatopoulos et al., 2016).

2.1.4 Manav (Dec 2014)

India's first humanoid robot, Manav is a 3D, two movement head that can nod and look around, the waist has 1-degree allowing waist movement like humans. It was designed in 2 months.

Table 2.2 Body Parameters of Manav

Parameter	Value
DOF	21
Tall	2 feet
Weighs	2 kg
Characteristics	21 sensors, 2 mikes, 2 cameras on head and eye

It has sound processing, visual processing which helps in responding to commands. Manav can walk, talk, and dance according to human voice commands. It can perceive depth and perception by binocular vision processing. Wi-Fi and Bluetooth are used for communication. The rechargeable lithium-polymer battery used by Manav can drive at least for an hour when fully charged. Manav responds like a human child.

2.1.5 ASIMO

ASIMO, humanoid robot structured and designed by Honda(Motor, 2007). ASIMO has a rechargeable 51.8V lithium-ion battery which has one hour working time. The robot can recognize moving articles stances, expressions, sound countenance, and a dynamic environment that encompasses them and connect with humans.

Table 2.3 Body Parameters of ASIMO

Parameter	Value
DOF	34, each leg has 6 DOF, each arm has 7 DOF, has 2 DOF for 4 fingers to grasp the object
Tall	130 cm
Weighs	54 kg and can carry a payload of 1kg
Characteristics	21 sensors, 2 mikes, 2 cameras on head and eye socket

The visual data grabbed with two eyes of camera situated in the head also evaluates separation and control of object approaching. ASIMO deciphers voice orders, human motions, recognizes when a handshake is offered or waved or pointed, and respond accordingly. It can confront an individual when addressed or look towards a sound. It can sense obstacles in front or rear and act accordingly.

2.1.6 iCub

iCub was structured by Robot Cub Consortium and assembled by the Italian Institute of Technology. Cub stands for Cognitive Universal Body. The motivation was human comprehension like a child learns by collaborating with its environment. Its robotic platform is adopted by 20 laboratories worldwide for research and academic development of robotic projects. It is an open cognitive robotic platform. iCub can see and hear and has sensing capabilities that help in body configuration and movement(Frank et al., 2014).

Table 2.4 Body Parameters of iCub

Parameter	Value
DOF	53
Tall	105 cm
Weighs	20.3 kg
Characteristics	It has attached 53 motors to control the movement of the head, legs, waist, arms, and hands

The product library is written in C++, utilizes YARP for outside correspondence using Gigabit Ethernet. It was not intended for independent activity thus doesn't have onboard batteries or processor however utilizes an umbilical link for force and system network connection.

iCub can crawl utilizing visual direction with an optic marker on the floor, tackle complex 3D mazes, facial expressions express emotions, grasping small objects, collision avoidance within a non-static environment, archery. To compute the physical interaction of a rigid body with environment and object, the iCub simulator uses an open dynamic engine. An open dynamics engine is a reliable physics engine, computing physical interaction between objects and the environment (Tikhanoff et al., 2012.).

2.1.7 POPPY

Poppy is a robust, reliable, and accessible, and 3D printed Humanoid robot. The open-source software and hardware of robots allow programming and experimentation of various robotics morphologies. The behaviors of Poppy were partially dependent on body configuration and controlled using a pre-wired electronics circuit. The Python programming controls hardware. The joint motor of Poppy controls its physical interaction. The design of Poppy is modular and can be easily modified and adapted to particular needs(Lapeyre et al., 2015). It is used by many schools and universities for exploration and research. The ideal medium of STEM.

Table 2.5 Body Parameters of Poppy

Parameter	Value
DOF	25
Tall	83 cm
Weighs	3.5 kg
Characteristics	One LCD screen, 2 wide camera attached on its head.

2.1.8 Romeo

The main aim of the development of Romeo humanoid was to assist an elderly person suffering from a loss of autonomy. The Romeo humanoid interacts with humans, roaming around human's physical environment, and helping in their needs. Romeo is capable able to extract a realistic perception and interact in -natural way with individuals(Pateromichelakis et al., 2014).

Table 2.6 Body Parameters of Romeo

Parameter	Value
DOF	37, a spine with 4 vertebrae
Tall	140 cm
Weighs	40 kg
Characteristics	During development, special attention was on facial movements, voice, and gestures, to increase effectiveness between human-robot interaction

Different layers of perception have been analyzed in Romeo from sense to interaction. Sensor and their fusion build a 3D point cloud world. Relevant information gathered from the 3D point cloud help to learn Romeo to categorize emotions and instructions(Pandey et al., 2014). Humanoid responds by natural speech and gestures to perform tasks like closing trash can, cup lifting with four fingers hand, and retrieving food from the kitchen.

2.1.9 PETMAN (Protection Ensemble Test Mannequin)

PETMAN, an anthropomorphic humanoid designed to test protection against chemical warfare agents. The embedded chemical sensor inside the skin of the

PETMAN humanoid robot measures and detects chemicals in the suit under controlled temperature and wind conditions. It maintains dynamic balancing when pushed moderately from the side. PETMAN is a free-standing bipedal robot powered by hydraulic power separately and its speed is about 7.08 km/hr(Nelson et al., 2012).

Table 2.7 Body Parameters of PETMAN

Parameter	Value
DOF	29, sensors for measuring position and force
Tall	175 cm
Weighs	80 kg, can carry a payload of 23 kg
Characteristics	Onboard control systems sensing, computing, and control movement. One passive DOF in each wrist and foot provide compliant interaction with the environment.

The robot gives reasonable tests during physical efforts, for example, controlling temperature, dripping, and dampness inside the protective apparel. It performs different undertakings powerfully in rising circumstances, for example, rescue operations in a fire, atomic, and different risky conditions without human introduction.

2.1.10 NAO (Aug 2008)

An approachable design of the NAO humanoid robot had been kept purposely. Due to the huge number of degrees of freedom, It offers great mobility. The open-loop engine controls the walking of the robot. The absence of feedback arises instability in NAO's movement. The open-loop stabilizer controls the motion of straight walk and follows arc without falling on the flat and hard ground(Shamsuddin et al., 2011). It is accessible as an examination robot for schools, universities, and colleges to train to program and execute exploration of human interaction.

NAO incorporates four amplifiers for recognition of voice and limitations of sound and two speakers for content to discourse union, 2 HD cameras for the exterior, and body recognition(Shamsuddin et al., 2012). NAO uses a Linux based operating system. NAO's programming is done in C++, Python, Java, MATLAB, C, etc. It also uses Ethernet and Wi-Fi for communication.

Table 2.8 Body Parameters of NAO

Parameter	Value
DOF	11 DOF in the lower limb, 14 DOF in the upper part of the body.
Tall	57 cm
Weighs	5.5 kg
Characteristics	The special pelvis kinematics design requires one motor which can bend forward without movement of legs.

2.1.11 Actroid-SIT (2003)

Actroid, the humanoid robot looks and does move in the direction of an individual attempting to address her. Actroid has a women-like figure and possesses 47 DOF. Out of 47 DOF, 29 DOF is provided for body gesture control and the rest are provided for facial expression control. Since her joints are controlled by pneumatic actuators, it has very little chance to get damaged. The flexible gestures in real-time can be generated due to a reconfigurable motion database. The reconfigurable movement database of Actroid has two fundamental highlights: motion interruption and its parameterization. At whatever point there is a disappointment because of speaker interference, it ends the social undertaking and then switches to the following response. A human-like movement grouping is acquired from the movement catch framework(Kondo et al., 2013).

Artificial Intelligence offers the capacity to respond alternately to an increasingly delicate activity like a pat on arm. It lacks locomotion either seated or standing. Speech recognition software and verbal responses through speakers. The robot can react in a restricted manner to non-verbal communication and manner of speaking by changing its outward appearance, position, and vocal inflations.

2.1.12 Bipedal/ Humanoids of 2020

According to the American Society of Mechanical Engineers (ASME), 10 humanoid robots of 2020 are Robotic Avatar, Robotic Ambassador, Delivery Robot, Research humanoid (Surena IV), Digital humanoids, Robotic Bartender, Robotic Actor, Robonauts, Educational Robot, Collaborative humanoids.

Sophia the smartest robot which uses the most advanced AI technology with a good sense of humor in the world. She is the world's first robot citizen. ASIMO is the most advanced and famous social robot and is continuously developing. MONONOFU is the world's largest robot, ROBOBEE is the world's smallest flying robot, the strongest industrial robot arm is M-2000iA/2300.

Total there are 3,053,00 units of the operational robot as per 2020 worldwide stock.

2.2 Bipedal Robot Motion

Bipedal walking in a humanoid robot is a complex project. Locomotion problems are because of a large number of degrees of freedom coupled with non-linear dynamics. The leg movement of the human body is controlled with the assistance of a biological rhythm called the central pattern generator (CPG). The central pattern oscillator controls the bipedal movement and whole humanoid body motion. In an unstructured and unknown environment, a humanoid robot is unable to control the leg movement.

Projects involving bipedal robots generally study the balancing and locomotion mechanism in a certain environment for applications where wheeled robots are completely not suitable. Most research has concentrated on getting a robot to stay stable when walks straight in a line. Along with the definite motion, the bipedal robot requires to explore the real uncertain world, by turning around, lifting one foot, moving sideways, stepping backward, and these movements issues involved are different from linear walking. This part of the literature review deals with the movement of bipedal.

(Lim & Yeap, 2012) have described the movement in bipedal considering human characteristics. Six servo motors, acceleration sensor, lithium-polymer battery pack, and remote control is used in the bipedal robot. Servomotor for joint movement receives signals from the wireless remote controller. Hence six push-button switches control the movement of feet. Several experiments on hardware were conducted to get the values for the correct posture of the locomotion. These values were given as input to the software further advanced developments were done.

(J. Park, 2007) used a pattern-based walking planner for ZMP control. A pattern-based walking planner creates a path of design variables such as direction and velocity. The desired motion of the pattern was generated by pattern control which is controlling COM and horizontal angular movement. Joint servomotors use inverse kinematics. These pattern controllers keep track of the desired trajectory of bipedal.

(Ken'ichiro, 1997) described an evaluative genetic algorithm and neural network controller. The camera captures the real-time visuals and simultaneously generates guided swing motion for the bipedal robot. The neural network controller uses a multi-layer preceptor, consist of four layers. The first layer (input layer) comprises two neurons, the second layer (middle first layer) comprises four neurons, the third layer (middle second layer) comprises four neurons and the fourth layer (output layer) comprises one neuron. Synaptic weights and thresholds value are optimized by the execution of the genetic algorithm, which usually is real values. Simulation is carried out in the virtual environment considering to be noise disturbance. Execution of the program was done on parallel computers. The bipedal performs the task by decoding the gene of the best individual as output this acts as input to a multilayered neural network designed.

(Yang et al., 2006) described the generation of steps for various sorts of ground which uses truncated Fourier series formulation. By adjusting ankle-pitch and knee-pitch angles of both the feet, the stability of bipedal is maintained. To avoid tooling, a zero-moment point criterion is used for evaluating generated gait given in joint coordinates. The genetic algorithm helps in maintaining the zero moment point to be within the footprint.

(S. C. Y. Kim & Hutchinson, 2008) proposed a hierarchical planner based on the workspace decomposition. The workspace decomposition consists of a passage map, gradient map, obstacle map, navigation map, and a local map. The greedy hierarchical planner algorithm is used to plan the motion of the bipedal. The workspace decomposition and connectivity graphs are stored in the data structure. The hierarchical algorithm consisting of local plans, global plans, and sub-goals. The decomposition of 3-dimensional maps encoded into a 2-dimensional workspace. The navigation map computes the passage map and the obstacle map. The robot is moving only in the free space of the target environment.

(Niiyama et al., 2010) investigated the musculoskeletal movement in Athlete bipedal running. The kinematic data patterns and data of muscle activity are measured for the leg movement. The bi-articular muscles supply torques at knee and hip joints simultaneously. Activation of knee joints and hip joints motor command uses parse coding of activation method. The required muscle force is determined from the desired force. The human electromyography data extracted for the muscle activation and its pattern are used for activation of the athlete bipedal robot.

The hereditary calculation balanced CPG boundaries to create control yield near wanted directions. CPG adjusts to outer signs from the capricious condition. Yields of CPG come back to characteristic motions, on evacuating outside signs.

(Inada, 2003) introduced the Matsuoka neuron model in a central pattern generator (CPG) and investigated for the bipedal movement. The output of neurons generates target angles of individual joints. The neuron generates rhythmic oscillation. CPG is usually included in balanced activities like mobility/ locomotion. The trajectories of each joint are captured from the human movement. The genetic algorithm adjusted CPG parameters to create control output near wanted directions and path. CPG adapts to external signals from uncertain conditions. Outputs of CPG return to natural oscillations, on removing external signals.

(Thuilot et al., 2002) analyzed the conduct of the compass robot walk model of the easiest bipedal. Bipedal had two indistinguishable legs that were joined to the hip and mass is concentrated at the hip. In place of knee joints, the prismatic joints were attached to the lower leg. All the joints in the compass are passive, they do not require any external power source. The compass gait is composed of the swing and the transition stages. In the swing stage, the compass hip is fixed by the point of support of the leg on the ground. Another leg swings forward. In the transition stage, the support is transferred from one leg to another. The compass robot makes stable walking on inclined surfaces also.

(Nishino & Takanishi, 1998) discussed an algorithm for controlled movement of bipedal and added calculation module to process algorithm which helps in improving the generality of bipedal. They discussed the control method of driven joints. Dynamic walking of the humanoid is controlled by the non-linear spring mechanism. The spring mechanism consists of wires and motors. The rotatory encoder gives angle feedback to control the tension in the driving wire. Coordination of motor on both sides, allow humanoid to take a step forward. One motor controls the spring tension of one side and the other motor is controlling the joint movement of the other side.

(Inoue & Takanishi, 1999) developed a control method bipedal robot for dynamic walking. Control framework comprises fifteen Alternate Current servomotors and 16 DC servomotors controlled through boards. The three-axis moment is generated by the trunk. Yaw-axis actuator generates moment along yaw-axis which is attached near the neck of bipedal. The swinging of the upper limbs generates moments along the pitch and roll axis. The algorithm computes the motion of the trunk, upper limb, and lower limb arbitrarily. The control system compensates for the ZMP and yaw-axis moment.

(Komatsu, 2005) proposed a modified central pattern generator (CPG) method to control the motion and force between the leg and ground for unknown environmental conditions. The control architecture of the hybrid central pattern generator control method consists of three layers. The first layer is creating rhythmic motion for the legs. The second layer controls the forces indirectly between the ground and foot and the third layer control attitude of the hip. The rhythm generators consist of a neuron model and each neural model consists of four units of oscillators. The oscillator generates torque for individual joints.

(Reil & Husbands, 2002) describe an evolutionary algorithm that controls the stable bipedal movement in a straight line. To achieve this task no perceptive information is required. The recurrent dynamical neural network-based controllers are implemented to achieve the desired bipedal motion. The population parameters of the evolutionary algorithm are deciding the fitness and weight in the recurrent dynamical neural network. The weight of joint movement and time remains constant throughout the motion of bipedal. The fitness function depends on the minimum distance from the origin and does not allow the hip joint to go beyond a certain height to avoid bending (forward/ backward) in a bipedal robot. This increases the stability of bipedal.

(Kuffner, 2001) presents a heuristic safe route calculation algorithm for bipedal in snag arranged environments. The methodology is to assemble a set of achievable areas of stride by processing stride positions to an obstruction jumbled environment. The planner generates a sequence of footstep placement by taking input from the collision-free environment. The polygon-polygon intersection method is used to avoid the collision for safe navigation.

(Tlalolini et al., 2011) gave an optimal walking movement with flat-foot and foot-revolution. The methodology is embraced for figuring torque delivered in the various joints. The optimal trajectory depends on a reasonable set of parameters. A cubic spline trajectory and constraints have been added to find the optimal trajectory. The simulation was carried out for the bipedal robot walk with foot rotation and without foot rotation. They established the localization technique using stereo vision for a humanoid robot. They found that the stereo vision creates jerky motion while walking due to noise present in the environment. The feature-based approaches use depth maps for localization. The stereo vision system mounted on top of humanoids and obstacles present in the environment are being captured. Motion captures system creates the elevation map which provides localization to humanoid on the ground.

(Rostami & Bessonnet, 1998) considered instability during the bipedal movement for a single support phase (SSP). The approach minimizes the joint actuating torque which ensures stability. The controlled walking is achieved by less impact and non-sliding heel touch. The Pontryagin maximum principle is applied for computing optimal motion synthesis for joint trajectory. During the swing, the bipedal robot is very unstable, and to overcome this problem, smooth motion and less energy consumption have been computed.

(Copyright et al., 2007) proposed integrated motion control for walking, jumping and running. The integrated motion control generates real time-based motion pattern which is based on dynamics involved in the humanoid. The adaptive motion control method was implemented for controlling zero moment point and leg ground contact. The zero-moment point has no acceleration and the height of the center of mass remains constant. Constraints had been applied on vertical zero moment point trajectory and angular momentum. The simulation is carried out on dynamic simulation software of QRIO humanoid robot and testing is being done for walking, jumping, and running.

(Caldwel & Bowler, 1997) explore the structure of pneumatic muscle to reduce energy consumption Internal structure of the pneumatic muscle actuator is different from the conventional pneumatic actuator. The inner layer of the pneumatic muscle cylinder is made from rubber tubing and the ends of the tube sealed by two aluminum plugs. The mechanism of bipedal design consists of free motion at the hips and knees. The limbs of bipedal are constructed from steel and aluminum. The actuation is provided to the leg by antagonistic pneumatic muscle actuators.

(Grizzle et al., 2009) introduced the MABEL platform for studying the walk, run locomotion of bipedal robots. The main purpose of the platform is the development of a new feedback control system for running and walking on rough terrain. The second purpose is to create motivation for building a robot for technology outreach. The mechanical architecture of bipedal is a planar robot with five links. Two legs with the knee are assembled on the torso and the legs are terminated in point feet. A real-time computing and data acquisition system acquires the data from sensors. The software framework switching controller module and helps in controlling bipedal.

(Y. Kuroki et al., 2004) developed an SDR-4X small bipedal entertainment robot. This software creates a whole-body motion. The SDR software consists of a motion creating system and foot trajectory generator system. The upper body motion was edited and created by the motion editor creator and the lower body trajectory was created by a foot trajectory generator. Upper body motion is created by the motion designer by loading the music into the system. The dance steps are created with the help of a foot trajectory generator.

(Y. Huang et al., 2013) obtained adjustable step length and velocity during dynamic bipedal walking. The compliance of the joints is controlling the passive walking to obtain natural motion robot experiments. A kinematic coupling is used to keep the upper body centered between two legs. Bipedal walker moves on level ground resembling a real robot. By changing joint compliance, the walking pattern is observed. The step length and velocity control the natural dynamics of the walker.

(Garofalo et al., 2012) suggested a periodic walking motion for the spring-loaded inverted version of a bipedal robot. The controller architecture of the SLIP bipedal robot consists of an upper layer and lower layer control. The upper layer control is straightforwardly associated with SLIP and guaranteeing periodic walking design. The lower layer controller controls the force. The main function of the controller is to set up an interface between the elements of a real robot.

2.3 Mechanical Design of Humanoid Robot

The human body consists of bones, muscles, cartilages, and joints. Push and pull of muscles control the movement of the body. It is difficult to develop the muscular-skeletal system in humanoid robot/ bipedal by mechanical components. The goal of mechanical design is the development of humanoid/ bipedal robots resembling humans. This part of the literature review deals with the mechanical design of humanoid/ bipedal robots.

(Yu et al., 2014) presented the mechanical design of a humanoid robot. They also proposed the control architecture of the control system based on the multi-channel communication system. The mechanical design of both legs of the bipedal robot consists of six DOF. The design principle consists of symmetry of the body like human and high stiffness and lightweight. Link and the joint was fabricated by mechanical casting. The distributed control architecture was implemented to control the joint motor movement.

(Borst et al., 2007) developed a research platform for the manipulation of the two-handed dexterous arm of the humanoid robot. The workspace of 2m was considered when the arm of the robot was designed. The robotic arm has an anthropomorphic kinematic configuration which can easily grasp the object from both the hand. The mechanical design consists of the torso, arms, head, neck, and fingers. The table is a workspace for the robot. The dexterous arm has 14 degrees of freedom and the hand has 24 degrees of freedom.

(Kanehira et al., 2003) developed an advanced leg module for rough terrains and prevent tipping over to protect the damage of the robot/ bipedal body. The design of the advanced leg consists of six DOF. Three DOF is provided in the torso and the knee has one DOF and the ankle has two DOF. The upper leg and lower leg are considered to be of equal length 30cm and the ankle length is 9.1cm and the length of the torso is 12cm. The total weight of both legs is 17.2 kg and the dummy weight considered on the torso is 22.6 kg. To prevent tipping over, the design follows the cantilever type structure of the hip joint. The design mechanism for walking on rough terrain consists of a six-axis force sensor and rubber bushes.

(Oh et al., 2006) developed an android Albert HUBO bipedal robot that has a height of 137cm and a weight of 57 kg. The actuator of bipedal consists of gear and DC motor. Planetary gears are used for the finger joint to reduce the backlash and small errors. Finger and wrist movement does not affect the stability of whole-body motion. Harmonic gear is used for the arm movement and leg movement to maintain the system stability and joint position stability. The weight distribution can be done by keeping all the power source battery and controller on the torso.

(Iwata & Sugano, 2009) proposed the anthropomorphic design of Twenty-One human bipedal robots that provide supports to elder women while securing contact safety. The passive impedance mechanism used in the Twenty-One bipedal robot. The upper body is attached to the base of the Omni wheel. The concept design of the Twenty-One bipedal robot started from the setting of the task scene in daily life. The anthropomorphic design of the upper limb consists of the arm and trunk. The arm has seven degrees of freedom with one redundancy and the four-finger hand has thirteen degrees of freedom.

(Fukaya & Toyama, 2000) proposed the novel design of the humanoid hand of the TUAT /Karlsruhe humanoid arm. The hand of the TUAT /Karlsruhe consists of 20 degrees of freedom. The first four fingers of the humanoid arm are identical and each one consist of three joint of four DOF. Palm consists of two DOF and the little finger moves freely. The joint of the finger is driven by one actuator. A special mechanism of link rod pulls link plate and finger moves. One actuator drives all the joints of four fingers which are placed into or around the hand. When the proximal joint touches the object, the finger curls around the object, and the adjacent part is moved by link.

(Ogura et al., 2006) describe the development of a bipedal simulator and design of robot WABIAN-2 bipedal robot. Each leg of this bipedal robot consists of 7 DOF and 2 DOF to the waist. The target is on designing the lower body of bipedal. The aluminum alloy is used for the fabrication of the WABIAN-2 bipedal robot, so that body has low weight and more stiffness. The 3D -CAD model was used for the design of the bipedal robot.

(Akachi et al., 2005) presented the mechanical and electrical features of the HRP-3P bipedal robot. The mechanical and structural features of HRP-3P protecting the body of the bipedal robot against water and dust. The mechanical features include the height, width, and depth of the bipedal robot. The height of the HRP-3P humanoid robot is 160cm, width 66.4cm, and depth 36.3cm. The total degree of freedom is thirty-six. One of the unique structures is a cantilever type hip joint and it allows the robot/ bipedal to move the cross leg.

(Ha et al., 2011) presents the design method of open bipedal platform DARwIn-OP. The mechanical design of bipedal consists of 20 degrees of freedom. The center of mass lying between the center of the hip. Frames of a bipedal robot are hollow in structures so that anyone can put the sensor between the gap of the frame. The height and weight of the robot are 45.5cm and 2.8 kg respectively.

(Endo et al., 2008) describe architecture and evaluation of a limited degree of freedom of head of a bipedal robot for facial expression of the WABIAN-2 bipedal robot. To provide the emotional expression in the bipedal robot, the ankle joint movement along the yaw axis and the trunk joint along the roll axis are removed from the bipedal robot. A new head design was proposed for balancing the head. The lightweight and downsizing of the head are considered to mount on the body. The wires and torsion spring attached to the yaw of the eye. The lip is of spindle type and actuated by wires made by using springs.

(Lohmeier et al., 2009) developed the fast walking bipedal LOLA robot. The emphasis was given on improving weight to achieve good dynamic performances. The height of LOLA is 180cm and 55 kg in weight. The redundant kinematic structure of the leg is lightweight and allows for a natural and flexible gait. The active toe joint occurs momentarily before the swing leg comes in contact with the ground thus reducing the joint loading. The toe contact with the ground stabilizes the bipedal and allow forward movement.

(I. W. Park et al., 2005) developed the mechanical design of the KHR-3 bipedal robot without hand. Pulley belt mechanism drives the joint of the humanoid robot. Belt tension is maintained by changing the motor position. The hip joint of the robot was designed as a tube-type crossing structure. The internal part of the tube is almost hollow except for gear actuation assembly. All the frames of the bipedal robot are 2D in shape. The closed kinematic configuration of the robot provides more support when it supporting on the ground. If some position error occurs in the feet then a comparison of the error in the motor position and the joint motor is done.

(Ambrose et al., 1973) developed space humanoid Robonaut robot which works in the space environment to assist the astronaut. They focus on the upper body design of the Robonaut. The arm and finger of Robonaut offer dexterity and sensing. The hand of Robonaut's will fit into the astronaut's hand gloved. The hand has a total of 14 degrees of freedom consist of the forearm, wrist with 2 DOF, and the five fingers with 12 DOF similar to the human arm. The head, consists of two color camera provide virtual reality and depth perception.

(Yamasaki et al., 2000) proposed the basic architecture and design of the PINO humanoid robot/ bipedal. The PINO has twenty-six DOF, each leg consists of 6 DOF, each arm consists of 5 degrees of freedom, the neck consists of 2 degrees of freedom, and the trunk consists of 2 DOF. All joints are actuated by a 26 DC motor. The metal gears reinforce against the high torque.

(Tellez et al., 2008) introduced mechanical stage Reem-B in the field of assistance robots. The psychological capabilities of the Reem-B bipedal robot enable dynamic walking and association with individuals. The physical structure of the bipedal robot comprises feet and a middle body. Stiffness ought to be high but weight ought to be less. The robot had one hand having four fingers and 11 degrees of freedom in fingers.

(Tsagarakis et al., 2011) built lower body parts of cCub bipedal robot, the progression of iCub bipedal. New leg motion system considered in the cCub bipedal robot. The mechanical structure of cCub includes every leg that has six DOF, three DOF in the hip, one DOF in the knee, and two DOF in the ankle. The leg has an andromorphic kinematic structure comprise of the hip, thigh with the knee joint, calf with ankle joint.

(Gienger et al., 2002) manage the structure and control engineering of a bipedal robot. For human walking movement - pelvic turn, pelvic roll, knee and ankle cooperation, and relocation of the pelvis are significant determinants. Joint torques depend on move relies upon the development of pitch movement profile. Ankle joint actuated by two linear ball screw drives.

(Mohamed & Capi, 2012) built up a versatile humanoid robot for helping older individuals working at home and clinic. A visual sensor perceives items. Laser discoverer sensor connected to the lower body. Bipedal starts by moving towards the target object. The upper part comprises arms, gripper, and head. The entire upper body is appended on the wheel. It permits the arm to be checked and imagined simultaneously.

(Wyeth et al., 2001) describe the structure of the humanoid robot. Total CAD model and explicit motor and transmission chose for building mechanical structure and are under the progress phase. The mechanical dimension of the body of bipedal is taken considering the biomechanical property of the human body scaled to the tallness of 120cm. Self-sufficient bipedal comprise 23 DOF. Feet and abdominal of bipedal comprise 15 DOF and the remaining 8 DOF consists of upper body parts.

(J. Kim et al., 2012) proposed the advancement of biped walking of robot Robray. The main emphasis is on the advancement of the lower body thinking about two structures of bipedal. One design includes an exploratory platform to evaluate torque and force while walking. Another architecture considers consistent and flexible leg mechanisms to decrease high occurrence. The kinematic design includes each leg having six DOF, waist with one DOF, and ankle having two DOF. The extent of the development of the hip joint expanded by giving a count balance between the axis of the hip joint.

2.4 Control Architecture of Bipedal Robot

The controller design of the bipedal robot is very complex as the model includes inactivity and inverse computation. Second-order responses are usually loud. A powerful control framework needs to control bipedal movement along with the entire body of the bipedal robot. Various degree of freedom in a bipedal framework makes it unstable and incapable of performing the desired task in unstructured and dynamic condition. This part of the literature review deals with the controller architecture of humanoid/ bipedal movement control.

(Burghart et al., 2005) present the cognitive design of the humanoid robot. Cognitive design is a mixture of three-layered leveled structure and behavior explicit module. The control framework of the humanoid robot is detached into a different module. Each part has its product and equipment module. The foremost layer of cognitive control design comprises of sensor and motor. The data originating from the joint position sensor, force sensor, and tactile are passed to the next layer for the execution of a task. The second layer perceives a framework that has access to the database, where information is stored. The last layer inside recognition is organizing all components data in solitary methodology.

(Kanda et al., 2002) proposed a valuable methodology for implementing behaviors in bipedal to connect with individuals. The information got from intellectual investigations utilized for the design conduct of bipedal and acts as a manual to coordinate among individuals. To oversee execution request robot framework executes organized modules and episode controls successively. The connection between bipedal and individuals is dependent on the situated module.

(Rosenblatt & Payton,1992.) present a fine-grained layered control system for robot control. The fine-grained control design gives insightful control components through a piece of essential. The decision is made based on behaviors. Every unit of the system model gets contributions from different units and outer sources. After getting input, organize the process at an actuation level and produce a single yield. The interconnection between homogeneous units is missing in the structure network.

(Brooks, 1987) adopted a layered methodology control framework for autonomous bipedal. In this approach, the problem is decomposed into a parallel task. The robot control framework starts with accomplishing the most minimal level undertaking and never changes alter framework involves the zeroth level of control framework. The next layer of the control system builds and examines data from the low-level system to accomplish a new task. The low-level system is unaware of upper layer processing. The same processing is repeated to achieve the new task at the next higher level of the control layer system. Each processor controls the specific task and runs asynchronously.

(Ly et al., 2004) introduced distributed control engineering for accomplishing objective assignments for a bipedal robot. The control design is organized into three levels and maps useful highlights in equipment and programming modules. The control design comprises of task planning, task coordinating, and the task execution layer. In task level planning, task description is received and allocated the subtask by selecting multiple subsystem controllers of the robot. After the selection of a subsystem controller, actions are generated by task-level coordination for the execution level to achieve the desired task.

(Y. Wang & Butner, 1987) depict computer control design to advance the computational preparation of robot control. The proposed control design disintegrates simultaneously assigned tasks into the system. The processor gets data from the sensor and passes direction data to the robotic processor which is then interpolated by the robotic processors. The interfacing of the control framework and manipulator actuator is done by the interface card. Decomposition of the task parallel results in very little time to execute and hence reducing response time.

(Yoshihiro Kuroki et al., 2003) developed amusement applications in SDR-4X bipedal. The constant sensor-based versatile control is applied for the bipedal robot to control body movement on the harsh and unlevel landscape.

Balancing movement control builds adjustments of whole-body movement and produces an obstacle avoidance movement for the upper body. The tendency and body pose are determined with the assistance of an accelerometer and force sensor. The adaptive control system realizes deviation and controls the body posture and prevent from tipping down.

(Simmons & Apfelbaum, 1998) created task description language (TDL) to control a robot. TDL is an augmentation of C++. The assignment control design involves three-layered - planning layer, executing layer, and behavior layer. assignments are characterized in TDL. A class identifier such as objective, command, monitor, and exception is followed by an argument. TDL does not have a return value so that control is not returned until the next command has been taken care of.

(Khatib, 1987) proposed a framework for controlling end-effectors' force in the constrained environment. The two-level control architecture enhances the performance of the position and active force control of the robot manipulator. The real-time position and force control implemented in the operational space with obstacle programming systems. The end effectors equation is established in dynamic decoupling. For the stabilization of the redundant manipulator, the joint force and the dynamic behavior is identified.

(Mansard et al., 2009) presented a system for the execution of a stack of the task of bipedal to work in a collective environment. The stack comprises of undertaking definition and taking care of arrangements. The product system began with substances and their chart, control emphasis must be performed. An error related to each task is computed. The software scripting framework interface exists for handling tasks.

(Posadas et al., 2008) designed a portable and modular control architecture for controlling the mobile bipedal. A distributed blackboard communication was established between the mobile software agent. The proposed architecture reduced the temporal problem by separating the elapse communication time from the execution time. First, the architecture established offline communication to move the bipedal by real-time soft bus and then process the code. When offline communication terminated, the bipedal robot obtained the agents. The agent executed the task without any external communication.

(Erhart et al., 2013) developed an impedance control scheme for robot cooperative manipulation. The cooperative manipulation reduces the internal stresses on the joint. The kinematic uncertainties arise due to improper grasping of the object. The impedance control scheme computes the end effectors' trajectories and removes the kinematic uncertainties. The evaluated trajectory is compatible with the object's motion. During the manipulation task, the kinematic coordination is achieved by the closed-loop manipulator dynamics.

(Asfour et al., 2006) proposed hierarchically control architecture of a bipedal robot for household activity. Hierarchical control design had three layers structure - task planning, task coordination, and actuator-sensor levels. In the first level, tasks are recognized by the client and determine subtask for bipedal. The corresponding actions are generated by the task coordination level. The sensory-motor level executes tasks to achieve the desired goal by the bipedal.

(Feil-Seifer & Matarić, 2008) describe the novel control architecture of B3IA in an autonomous bipedal system for the behavior intervention of autism spectrum disorder children. The control architecture of the behavior-based behavior intervention architecture (B3IA) consists of the sensor and the interpreter module. This module helps the bipedal to observe and control the behavior of humans and objects in the environment. The operational decision is made by the bipedal in the task module. The operation of hardware is controlled by the effector's module. Human actions and bipedal actions are stored in the activity history module. The analysis and interaction of the robot with humans are evaluated in the evaluation module, which can be used as a bipedal parameter.

(Galindo et al., 2006) implemented the human–robot-integration control architecture into a robotic wheelchair. This control architecture permits a person to deliberate the activity. The human-robot control architecture scheme is made up of several elements. Modules are grouped into three layers. The hierarchical and symbolic representation of the environment is maintained in the deliberative layer. The internal world model is used to produce plans and establish communication between human-robot. In the execution layer, the sequences of the task executed and supervise the information received from the robot's sensor. The functional layer controls the navigation and manipulation between two-point and provides guidance.

(Naumann et al., 2007) deal with the control architecture of process-oriented programming for robot cells that enables in the production environment. The interconnector module of the software is taken as input descriptions and processes. The process-oriented programming is used to trigger the machining operation. To understand the process command, an ontology was introduced. Ontology is considered a relevant class of the robotic domain.

(Liu et al., 1989) proposed the adaptive neural network for robot hand control to grasp the object. The control architecture is based on the prehensile function of humans. The object analyzer module establishes the relation between object, shape, and grasping modes. The object analyzer module generates a suitable grasp mode for the robotic hand. The neural network generates the eight generic grasp mode for the robotic hand. This approach reduced the building of device-dependent grassing mode.

(J. Y. Kim et al., 2005) utilized distributed control engineering to control the joint of bipedal to decrease the computation time of the main controller. This controller is placed on the back of the bipedal robot, coordinates with the sub-controller at run time by the controller network area. The controller can receive and send data to the sub-controller at the same time. The sub-controller is designed separately for joint motor control and inertia sensor.

(Yokohama & Takashima, 2002) developed an open hardware platform for humanoid robotics. The virtual bipedal robot platform is compatible with a real bipedal robot as it is. The OpenHRP has the dynamics computation, contact and collision computation, and unification of the controller features. The OpenHRP is implemented on the common object request broker architecture(CORBA), which supports C++ and java programming. The simulation is controlled by a CORBA client in an integrated simulation environment.

(Rohmer et al., 2013) introduce virtual robot simulation platforms for the reconciliation of actuator, sensor, and control. A disseminated control strategy is utilized in an adaptable and scalable robot simulation platform. Three techniques were used to achieve the simulation. The first technique is the execution of the control code on another machine, so that computation time for simulation is very less. The second technique is the execution of the control code on the same machine for other processes than the simulation. The third technique is the execution of the control code on the same machine other than the simulation loop.

2.5 Reinforcement Learning Control Algorithm

The bipedal robot can understand the unstructured and unknown (dynamic) environment to perform the desired task. They just learn from the environment and execute the task without any human programmer or user. The reinforcement algorithm helps the bipedal robot to decide the critical condition. This part of the literature review deals with reinforcement learning in a bipedal robot. This is the current trend that is going on the developing the bipedal platform for the real environment.

(Weiß, 1995) discussed two distributed learning algorithms that are suggested: ACE (Action Estimation) and DFG (Dissolution and formation of groups). Learning achievements rely upon the exploration of an adequate number of state-action pairs. If the state, action spaces are enormous then it takes too much time for learning which is impractical.

(Zhou, 2002) proposed a genetic algorithm-based fuzzy reinforcement learning (GAFRL) agent who learns by using a global optimization technique can predict the capabilities of the critic network and evaluate the candidate solutions. Assumption of Fitness function incorporates various observationbased data to GAFRL agent and other machine learning methods for accelerated learning of robot.

(Pontrandolfo et al., 2002) utilized the RL approach applied to a case model, organized networked production framework that transverses a few geographic zones and different coordination stages. Fails as has not considered a huge and complex global SCM problem.

(Yen & Hickey, 2004) included a forgetting mechanism and used hierarchically structure RL agents to expanded execution when contrasted with traditional RL agents exploring in an uncertain environment. It is not practically implemented in Hardware.

(Ueda et al., 2004) acquired fingertip trajectories by RL dependent on simulation. The reward function has been designed by considering the friction between finger and page. This results in perfect turning of pages as there is no slip between the finger and the paper. It is required to achieve smooth movement of fingertips and on-line error compensation using visual feedback.

(Y. C. Wang & Usher, 2005) developed an application of agent-based production scheduling which utilizes RL algorithms to dispatching rules selection problems to determine whether it can be utilized for enabling learning of machine agents. It is not applied to complicated agent-based scheduling like dynamic job shop scheduling.

(Ling & Shalaby, 2005) proposed to automate street car grouping control utilizing multiple RL agents that follow up on the progression of progressive signalized crossing points. Multi-agent work in sync to separate road car group if one is recognized and to construct sensible progress between the matched road vehicles. It required deciding the optimal number of RL agents and the best settings and limitations of each agent. The development of state-space data to catch general vehicular traffic conditions on the major and minor streets may additionally improve the presentation of agents.

(Tehrani & Kamel, 2005) considered the Robot soccer problem for analyzing behavior arbitration. It utilized the Sarsa(λ) algorithm with a pseudo-fuzzy strategy for function estimation. Some underlying information is supplied to the RL learner. Team size considered was ONE learner on each side playing against each other

(Yasuda et al., 2006) applied RL that embraces the Bayesian discrimination strategy for sectioning persistent state space and consistent activity space at the same time. A real robot experiment was not carried out. It did not acquire more sophisticated cooperative behavior as the obstacle avoidance in the complex environment.

(Janssens et al., 2007) have done the allocation of neighborhood data in a simulation of activity-travel design. There is no limitation on the number of activities and consolidation of sensible travel time. The information does not reveal a significant relationship between time and the area.

(Duan et al., 2007) uses the Fuzzy Neural Network along with RL (FNN-RL). The residual algorithm is utilized to figure out the slope of the FNN-RL technique to ensure convergence and speed of learning. It uses a hierarchical learning method for robot soccer agents. The specific simulators FIRA 5 is required and the number of trails is 50.

(Kareem Jaradat et al., 2011) created a Sequential Q-learning algorithm to manage issues of behavior conflict that emerge in a multi-robot transportation framework. RL and GA are coordinated to settle on choices when the robots cooperatively transport an object to the goal location while staying away from snags. The sequential Q-learning gives good results for the sequential task but not for concurrent tasks.

(Nanduri & Das, 2009) introduced a computational algorithm to acquire Nash equilibrium of *n*-player matrix games. The algorithm utilizes a stochastic-estimate- based RL approach and can understand n-player network games with huge player–activity spaces. The proposed algorithm needs hypothetical evidence for convergence and optimality.

(Quintía et al., 2010) attempted to boost time before the robot fails to acquire a control strategy that is suitable for desired conduct. It does not consider the contrasts between what the robot predicts and what occurs in the real-time environment.

(Tamei & Shibata, 2011) introduces the utilization of policy slope kind of RL for conquering time-varying nature issue by formulating EMG-based active human-robot cooperative work as objective-oriented errands. Permitting increasingly broad 3-D movement, and utilization of the way to deal with progressively complex assignments such as motor learning and recovery.

(Shokri, 2011) suggested for each action of a very large state space, its associated inverse action is characterized. The state and its inverse action are defined in the structure of RL to refresh the value function which results in converging fast. The decision of the RL signal and inverse RL signal is critical. In certain applications, the inverse is not known inverse actions have to be reserved during the learning procedure.

(Zeng et. al., 1996) recommended a combination of RL and simulation to optimize operation schedules for the compartment terminals, which utilizes a simulation model to build framework conditions and is applied to learn optimal dispatching rules for various procedures. Designing reward function for rules and action procedures more effectively and contemplating the collaboration among various kinds of agents to improve coordination of the operation system.

(Maravall et al., 2013) examined the impact of group size focused on a group of the moderate size of the request of 5 and 10 people and the impact of the lexicon size on the convergence results. It has not used a physical multi-robot system.

(Gabel & Riedmiller, 2012) proposed that in the learning phase, agents adapt the parameters using the policy gradient RL, which aims to improve the performance of the joint policy scheduling objective function. A proposed lightweight communication system that improves agents' abilities beyond job dispatching. The policy gradient algorithm proceeds as stochastic gradient descent and the number of strategy refreshes required to reach a local optimum is expanding. They utilized a comparatively high estimation of E to acquire reliable gradient estimates.

(Matsubara et al., 2013) proposed a novel RL system for learning motor skills that communicate with flexible substances. Learning structure centers around the topological relationship between the configuration of robot and flexible substances when almost all details of the substance are considered(e.g. wrinkles) even which are insignificant for doing motor assignments.

(Velentzas et al., 2018) applied the reinforcement algorithm in assistive robots for the educational application. The child's gaze provides the information to the robot. The reinforcement algorithm has a set of the state which are dimensional features. The action has a finite discrete set of actions and generates a set of actions for the different states. The Q-learning rule helps to choose the action depending upon the task. After choosing the action, the transition takes place, and a reward is associated with the action and learns from history. The reinforcement algorithm decomposes the task into a set of discrete actions so that it can be easily understood by children-robot interaction.

(Katic & Vukobratovic, 2003) presented the hybrid control of an intelligent control system for the bipedal robot. The reinforcement learning accumulated the knowledge from the dynamic balance of the bipedal robot and improving the gait during walking. The control architecture of the gait synthesizer has three components. The neural network trains the action selection network using the error signal received from the external reinforcement. For the desired state, the action evaluation map state and failure into a scalar score. Stochastic action modifier uses the recommendation action and reinforcement to produce a dynamic walking.

(Kober & Peters, 2006) worked on episodic reinforcement learning to control the motor primitives in a dynamic situation. The policy gradient method is used in the reinforcement algorithm. For the desired control of motor both the dynamics of the system are chosen for the stable condition. The deterministic mean policy depends on the joint position and the basis function. The basis function is the motor primitive parameters.

(Kartoun et al., 2010) proposed collaborative interaction between humans and robots based on reinforcement learning. Learning is based on a collaborative Q-learning approach and provides a robot with self-awareness and autonomy. In a collaborative Q learning algorithm, there are two levels of collaboration between humans and robots. In the first level, the robot decides the action and updates its state-action values. In the second level of collaboration, the robot takes the request from the human advisors. The robot is switching from the autonomous mode to semi-autonomous mode based on the policies.

(Wawrzyński, 2012) demonstrate the reinforcement learning algorithm for bipedal gait optimization. The actor-critic learning applied for the experience replay and fixed point method to determine the step size. The Markov decision process provides the solution for the reinforcement algorithm to control the bipedal robot gait. The control process of actor-critic works in discrete time to select the state and select the proper action. The transition between the current state to the next state happens and a reward is assigned to the state and action. The stochastic control and the value function updates the learning parameter based on the data collected.

(Kati & Vukobratovi, 2006) proposed the fuzzy reinforcement hybrid control algorithm for the bipedal robot locomotion. The controller has two feedback loops around the zero moment point. The centralized dynamic controller keeps tracking of the robot's normal trajectory and fuzzy reinforcement feedback compensates the dynamic reactions of the ground around the zero moment point. Fuzzy reinforcement control algorithm structure based on the actor-critic temporal difference method. The policy represents the set of control parameters.

(Frank et al., 2014) proposed reinforcement learning which could control the iCub humanoid robot. iCub learns from its experience the world model and controlling actual hardware in real-time with some restrictions. Reinforcement learning discretized the real configuration of the robot in configuration space.

The modular behavior environment of the iCub humanoid robot generates the action and the robot tries to go into the transition state. The Markov model develops the path planner and connects the state to the near state.

(Christen & Stevˇ, 2019) proposed the deep reinforcement learning algorithm to train the control policies for the bipedal robot interactions. The control problem is formalized from the Markov decision process. Input to control policy is a joint position, velocity, and sensor reading of the hand. The motion capture system captures the position of the hand. The output of the control policy actuates the humanoid arm. The reward is provided to correct the end configuration of the humanoid arm.

(Lober et al., 2016) improve reinforcement learning using Bayesian optimization for whole-body motion control. It evaluates the cost function in robotics and optimizes the set of parameters. To ensure a smooth trajectory, the whole-body control guided by the task in a series of waypoints. Three components of costar are evaluated for the execution of the task. The optimization variables are selected from the trajectory waypoint.

(Hester et al., 2010) describe a model-based reinforcement algorithm with a decision tree to train the bipedal robot to kick goals. In the model-based reinforcement algorithm, learning takes place aggressively during model learning. Q-learning approach adopted for the model-free reinforcement learning. Q-learning update state-action for every state-action pair. RL with the decision tree take the action with the highest value and entering into a new state. After entering into a new state, the award will be received in the new state. Observing new experiences through the model, the algorithm updates the parameter through the model.

(Riedmiller et al., 2009) work on the application of batch reinforcement learning in a challenging and crucial domain. Reinforcement learning helps the robot to gain ideas from the repetitive interaction from the environment. The batch reinforcement control algorithm consists of sampling experience, training and batch supervised learning. The training pattern set estimates the value function. The batch supervised generates a new estimate for value function from the training set pattern. The behavior-based approach is used to implement the reinforcement algorithm to take the decision.

(Iida, 2004) proposed an adaptive allocation method for the reinforcement control algorithm for bipedal motion control. Actor-critic learning is adopted for reinforcement learning. This method has separate memory to represent policy i.e. independent from value function. The actor calculates the action value for the bipedal robot when it observes the state in the environment. The critic receives the reward and provides the temporal difference. The learning is simulated on the virtual body of the bipedal robot to stand up from a chair. The bipedal observes the wait, knee, ankle, and pitch angle of the body. The humanoid robot learns to fall backward. Afterward, it falls forward. Finally, it stands up and controls its body.

(Sko, 2008) proposed the dynamic control approach for the humanoid bipedal walking. The controller involves two feedback loops. The computational torque controller receives the input from the impact force controller and reinforcement controller. The reinforcement controller maintains the torso movement with the help of fuzzy feedback. The policy gradient reinforcement learning controls the trajectory of the dynamic walking of the bipedal robot.

(Danel, 2017) proposed actor-critic neural network architecture for the continuous action policy of reinforcement learning. The deep deterministic policy gradient method controls the bipedal body control task. The deterministic policy was developed by the actor-network. The action value generates by the critic network. The temporal difference was minimized by the training of the critic network. The immediate reward is received upon the action and update the learning parameter to the database of the control architecture.

(Peters et al., 2003) discussed the different approaches for reinforcement learning algorithms for a bipedal robot. The natural actor-critic learning controls the motor of bipedal. The movement plan has a set of joint position and joint velocity of the bipedal. The system has point-to-point continuous movement i.e. the episodic task of the reinforcement algorithm. The evaluation of the basis function for the value is done by the actor-critic network.

(Guenter et al., 2007) developed an algorithm for programming robots by demonstration. When unexpected perturbations occur, a robot is unable to perform a task. The teaching of a constrained task to a robot by a learned speed trajectory. The natural actor-critic network evaluates the policy by approximating the state-action values. The simulation is carried out for the cubic box and obstacle. Using reinforcement learning, the system takes 330 trials to achieve the goal.

(Lowrey et al., 2018) developed the control policies in a simulation that can transfer to the dynamical physical system. The policy gradient learning method is used in the reinforcement algorithm to optimize the parameters. The natural policy gradient algorithm pushing the task to learn. Training of policy determines an action to take and gain a good reward. The structure of training informs the policy behavior with the time required to execute the desired task. The reward function reduces the gap between the robot and the target.

(Kober et al., 2011) developed the reinforcement algorithm which maps circumstances to meta parameters. Motor primitives are used for learning meta parameters. The dynamical movement of the motor is represented in the first-order differential equation for the critical damped. The goal parameter is the function of the amplitude parameter that represents the complex movement. All degrees of freedom of the system synchronize in the dynamical equation in the canonical form.

(I. J. Silva et al., 2017) proposed a reinforcement learning algorithm to optimize parameter values for the generation of gait patterns in bipedal. Locomotion control achieves by the central pattern generator. The three oscillators attached to the foot of the bipedal. Each oscillator has six sub-oscillator related to the axis and configured to the parameters. The parameters are divided into three groups of offset parameters, oscillation parameters, and feedback parameters. Two parameters have been selected for the optimization of the gait.

(S. K. Kim et al., 2017) demonstrated the intrinsic interactive reinforcement learning algorithm for human-robot interaction based on the gesture posture. The human electroencephalogram generated feedback used for the reward. The leap motion controller recognizes the human gesture to learn the robot and simultaneously the robot map the gesture for action. The contextual bandit approach is used to enable the robot's action provided by human gestures.

2.6 Research Gaps

After a detailed literature review, the research gap was found out for the humanoid/ bipedal robot. Issue despite everything exists in the mechanical structure, movement, and controlling of the Bipedal robot. Some of the gaps identified are :

- All available systems either use Artificial Neural Network (ANN) or Artificial Intelligence (AI) for Robot and Humanoid/ Bipedal to design them for specific tasks/problems.
- The systems defined usually have predefined problems, conditions (environment), and pre-stated results to define them.
- Self-awareness is a humanoid robot that is missing to decide an unknown environment.
- Stability in the motion of a humanoid/ bipedal robot isn't accomplished completely in the dynamic conditions and uneven ground.
- Speedy walk and run in a humanoid robot are questionable. The abrupt turning of movement is an additionally unsolved issue.

The gaps taken into considerations are designing the framework for bipedal which used the reinforcement learning algorithm, the environment is dynamic as an object is placed at different locations and the targeted output is the smooth trajectory of the bipedal in the unknown and dynamic environment, the stability is a major concern. The locomotion is considered on a smooth surface, bipedal learn the walk and then executes which gives bipedal decision-making characteristics to whether explore more states of the dynamic environment or exploit the previously used states for fast execution to reach the goal in the minimum amount of time.

2.7 Research Objectives

The main objective for the proposed work is the design and simulation of the framework for a Bipedal walking robot (Finite State Machine) using a reinforcement learning algorithm.

The sub-objectives to fulfill the main objectives includes designing and simulation of algorithms which has the following features:

1. **Incorporating of forgetting mechanism into the RL system-** An agent might use the knowledge that has become outdated.
2. **Use of feature-based state knowledge in RL system-** For reducing the number of state values to be maintained, and
3. **Hierarchical organizing of RL system-** for reducing complexity in many applications

The result of the above objectives will be MAS developed which would answer two basic questions:

i. How can multiple agents learn which actions must be done simultaneously?

ii. How can multiple agents learn that all sets of simultaneous activities must be done consecutively?

The success of learning depends upon the exploration of an ample number of state-action pairs.

The robotics research community tries to implement human thinking behavior in the bipedal robot, which helps in the understanding environment and in deciding critical conditions. The reinforcement learning algorithm allows the bipedal to learn, think, and do the action in an unknown environment. The development of a reinforcement algorithm for several degrees of freedom is a very challenging task. Therefore, this research work aims to develop a reinforcement learning algorithm for vision and motion control in a bipedal robot by keeping the best options for future use.

2.8 Research Outcomes

This architecture is simulated in SimSpace Multibody implementing a Forgetting Q-learning algorithm and Feature-based Object Identification reinforcement learning algorithm to be implemented on the control system of the bipedal walking robot in a hierarchical structuring manner. The Forgetting Q-Learning algorithm is implemented on a hierarchical structure i.e. Hip joint is trained first then the knee joint is trained and then the ankle joint of one leg then after a delay of half of Gait time another leg joints are trained similarly to complete one Gait or Stride. The self-learning of bipedal to balance at runtime /online and then navigate to the identified object in the dynamic environment.

CHAPTER 3 PROPOSED MODEL

The proposed model considers lower body segments of bipedal. The model comprises ten degrees of freedom with every leg having five degrees of freedom. Both ends of the legs are linked to the torso. The torso is an inflexible body on which both hip joints are attached and have a vision system. Simulink/ SimSpace Multibody Matlab model is designed for the bipedal. For designing and developing the lower body of bipedal, the anatomy of the human lower limb (hip or pelvis joint, thigh, knee joint, calf, ankle joint, and ground contact forces) is taken into consideration(Agarwal et al., 2015). Table 3.1 shows the lower body parameter of bipedal without considering the upper body parts like shoulder, hand, and head(Sharma et al., 2020).

Table 3.1 Lower Body Parameter

Parameters	Dimension in mm
Foot length	240
Foot width	90
Foot height	100
Lower leg height	380
Lower leg diameter	370
Upper leg height	380
Upper leg diameter	480
Torso length	330
Torso width	150

The human body is designed to support the skeletal system. The bones of humans are rigid but cartilages make the body flexible. The appendicular skeleton of a human includes bones of the shoulder girdle, upper limbs, pelvis girdle, and lower limbs. The pelvic forms a supportive framework for the lower body.

The biomechanical factor is considered for the design of lower parts. The lower body system of the bipedal robot consists of the left leg, right leg, and torso. The kinematic configuration includes the degree of freedom of the joints, motion ranges of all joints, and length of links related to the motion of the lower body system. The movement of all joints together at a given point should be such that the motion of each joint should not restrict the motion of the other joint (Cenciarini & Dollar, 2011). Table 3.2 shows the degree of freedom of different joints and possible joint range of motion(Hernández-Santos et al., 2012).

Table 3.2 Joint Range Degree of Freedom and Motion

Joint		Standard Human Leg (in Degree)	Proposed Humanoid Leg (in Degree)
Torso	**Pitch**	-15 to 130	-15 to 100
	Yaw	-45 to 50	-45 to 45
	Roll	-30 to 45	0 to 45
Knee	**Pitch**	-10 to 150	0 to 100
Ankle	**Pitch**	-20 to 50	-20 to 30

3.1 Simulink Model of Bipedal Robot

While designing various parts of the bipedal robot, plane consideration was taken into account. Torso and hip joint created in the frontal plane of a sketch. The rest of the part is created in the sagittal plane of the sketch. In the actual model, the diameter remains constant throughout the height of the lower leg and the upper leg. The torso is kept fixed for assembling all the parts such as the pelvis joints of both the limbs. Torso also contains a camera mounted on it for object identification and vision-based navigation.

Figure 3.1 shows the complete 3D model of the lower parts of the bipedal robot without the ground.

The ground of the proposed model is created in the MATLAB SimSpace Multibody toolbox. The SimSpace Multibody toolbox provides the simulation environment for the proposed system. Figure 3.2 shows the MATLAB Simulink model of the lower body of bipedal without the ground.

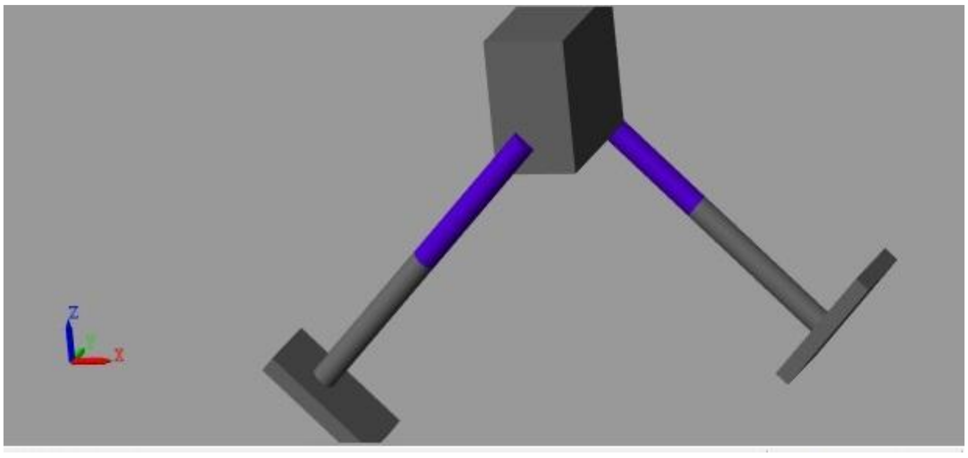


Figure 3.1 Proposed 3-D Model of Lower Body

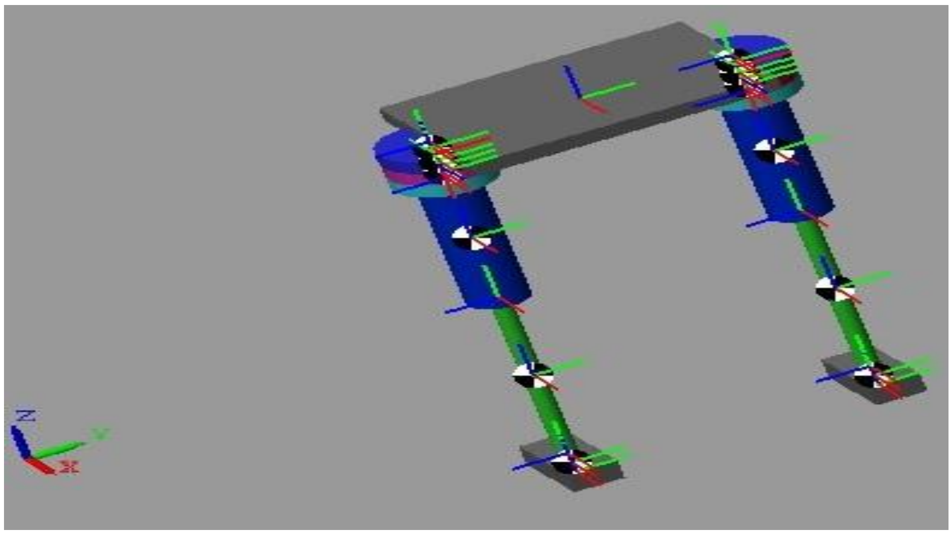


Figure 3.2 Simulink Model without Ground

Figure 3.3 - 3.5 shows the joint, frame, and rigid body MATLAB block representation of the lower body of the bipedal(Sharma et al., 2020). The complete combined model is shown in Appendix A.

3.2 Simulink Model of Bipedal with Ground

The Simulink model of the lower body of the bipedal with the ground is displayed in Figure 3.6.

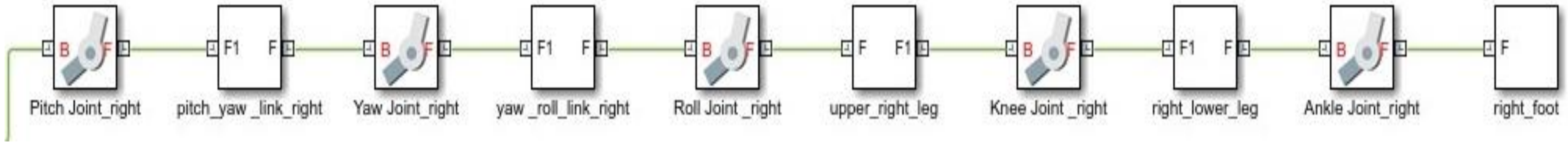


Figure 3.3 Simulink Sub-Model of Right Leg of Bipedal

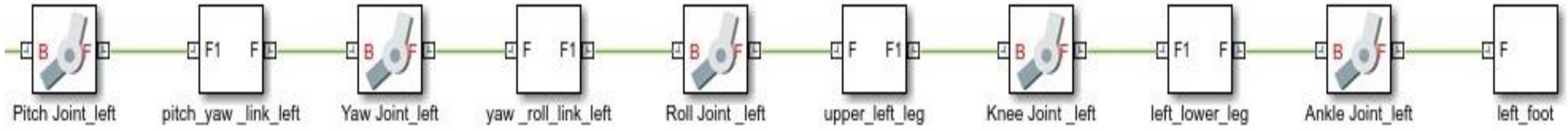


Figure 3.4 Simulink Sub-Model of Right Leg of Bipedal

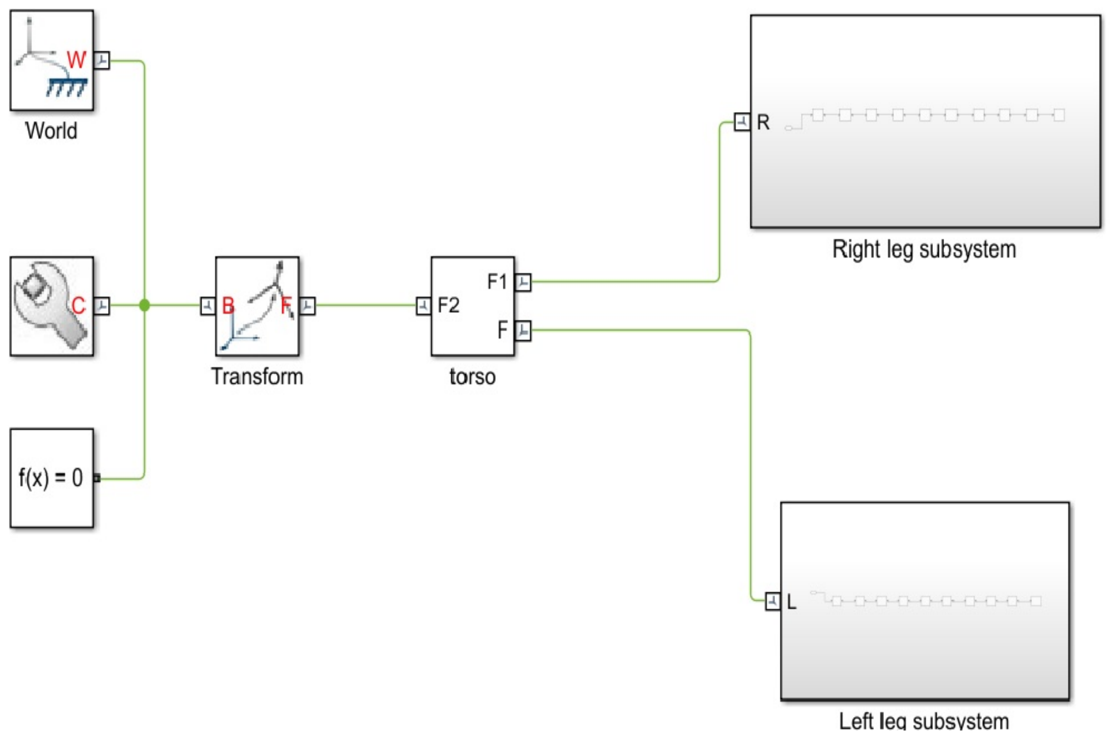


Figure 3.5 Overall Simulink Model of Lower Body of Bipedal

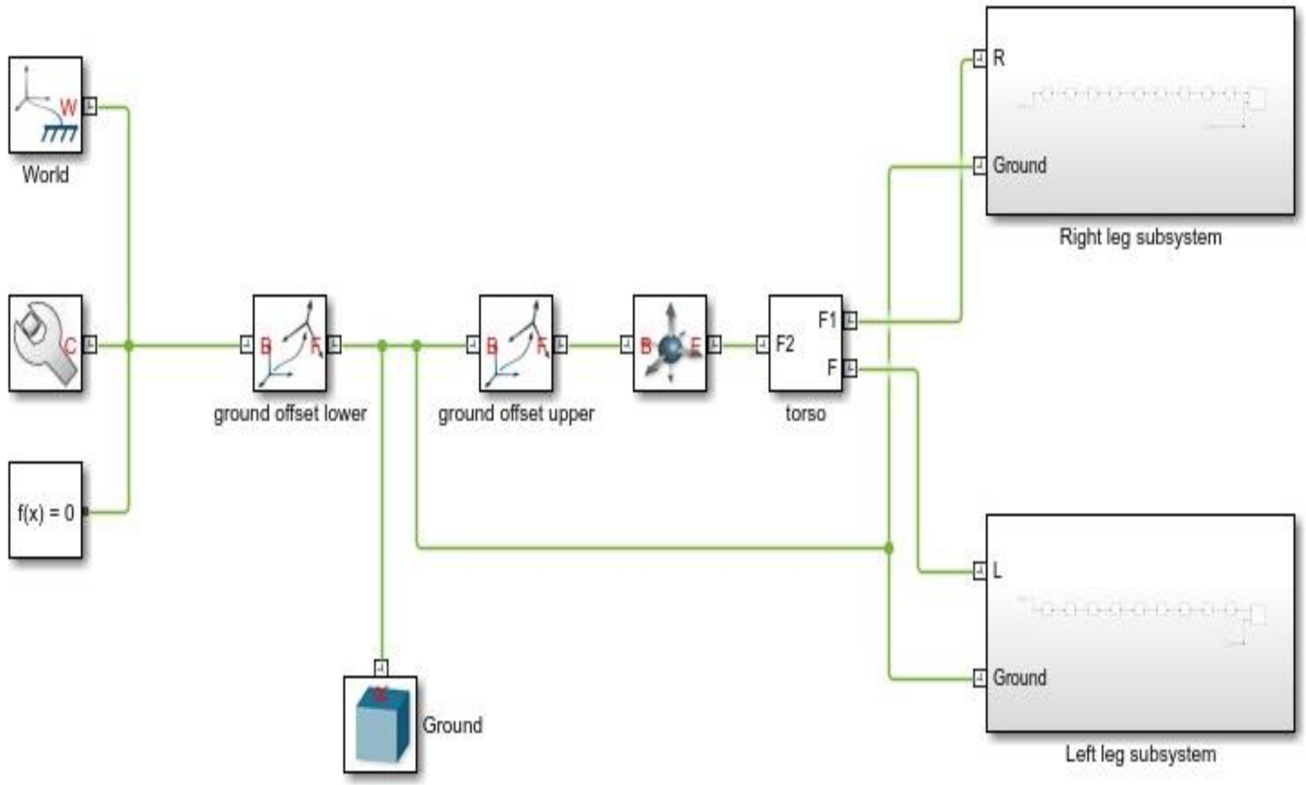


Figure 3.6 Simulink Block Diagram of Lower Body of the Bipedal with the Ground

The Simulink model of the lower body of the bipedal generated by the explorer of MATLAB is shown in Figure 3.7. Appendix C gives a representation of the mathematical expression of dynamic torque in the MATLAB Simulink model.

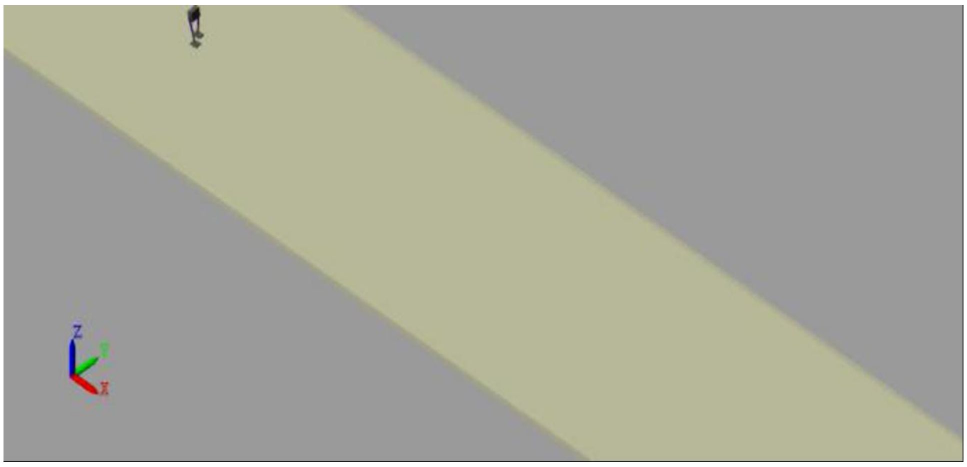


Figure 3.7 Simulink Model of Lower Body of the Bipedal with the Ground

3.3 Simulink Model of Bipedal with the Ground and the Contact Forces

The feet of the lower body of the bipedal applying forces on the ground. The friction force between ground and feet helps bipedal to walk. For creating the ground in MATLAB Multibody toolbox, a brick element of body block is chosen. For each body block, two rigid transform blocks are required (M. Silva et al., 2015). The tangential force empowers bipedal to make a forward motion on the ground. The normal force ensures that bipedal will consistently be above ground during the simulation. Figure 3.8 and Figure 3.9 show the Simulink block for ground and contact forces of foot interaction with the ground.

After calculating the different forces, the Simulink model of the ground is made with the help of different Simulink blocks. The detailed parameter of the ground block, rigid transform, and bipedal parameters are given in Appendix C.

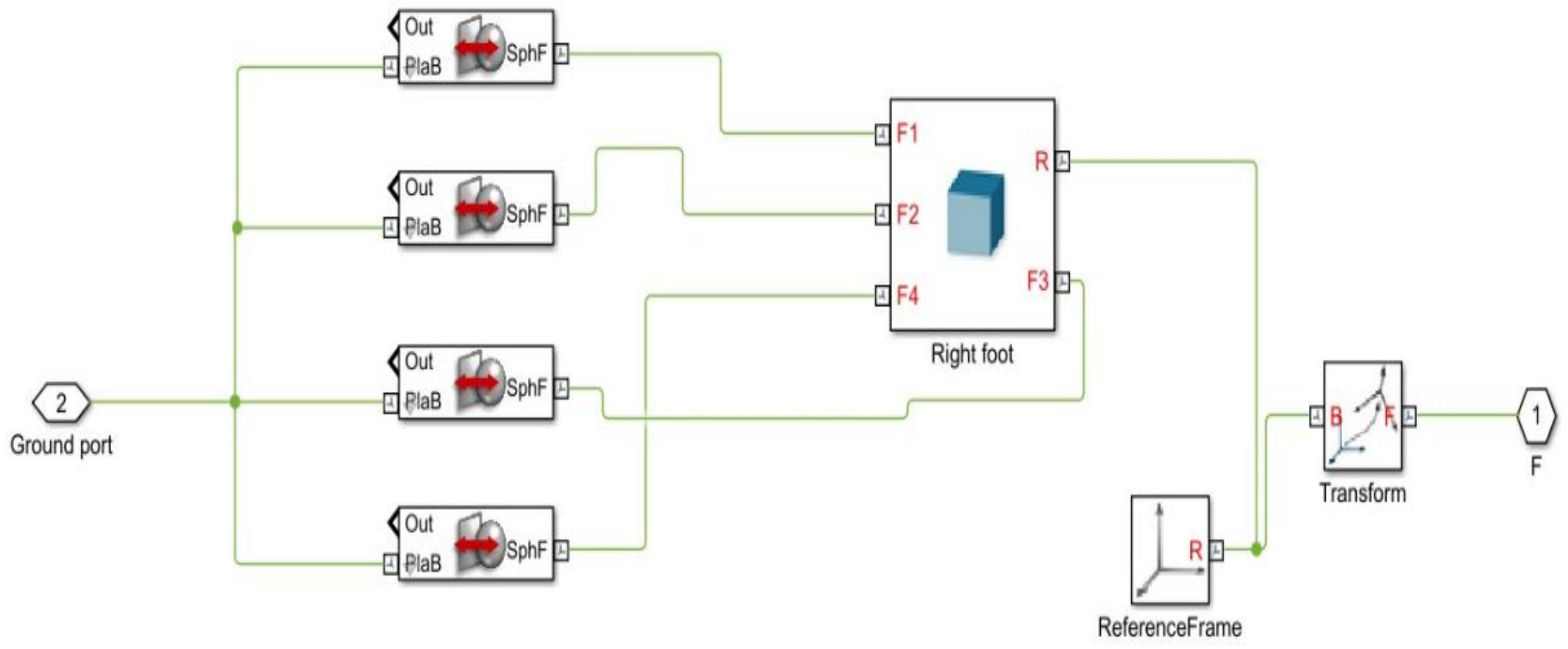


Figure 3.8   Simulink Block Diagram of Contact Forces of Right Leg of the Bipedal with the Ground

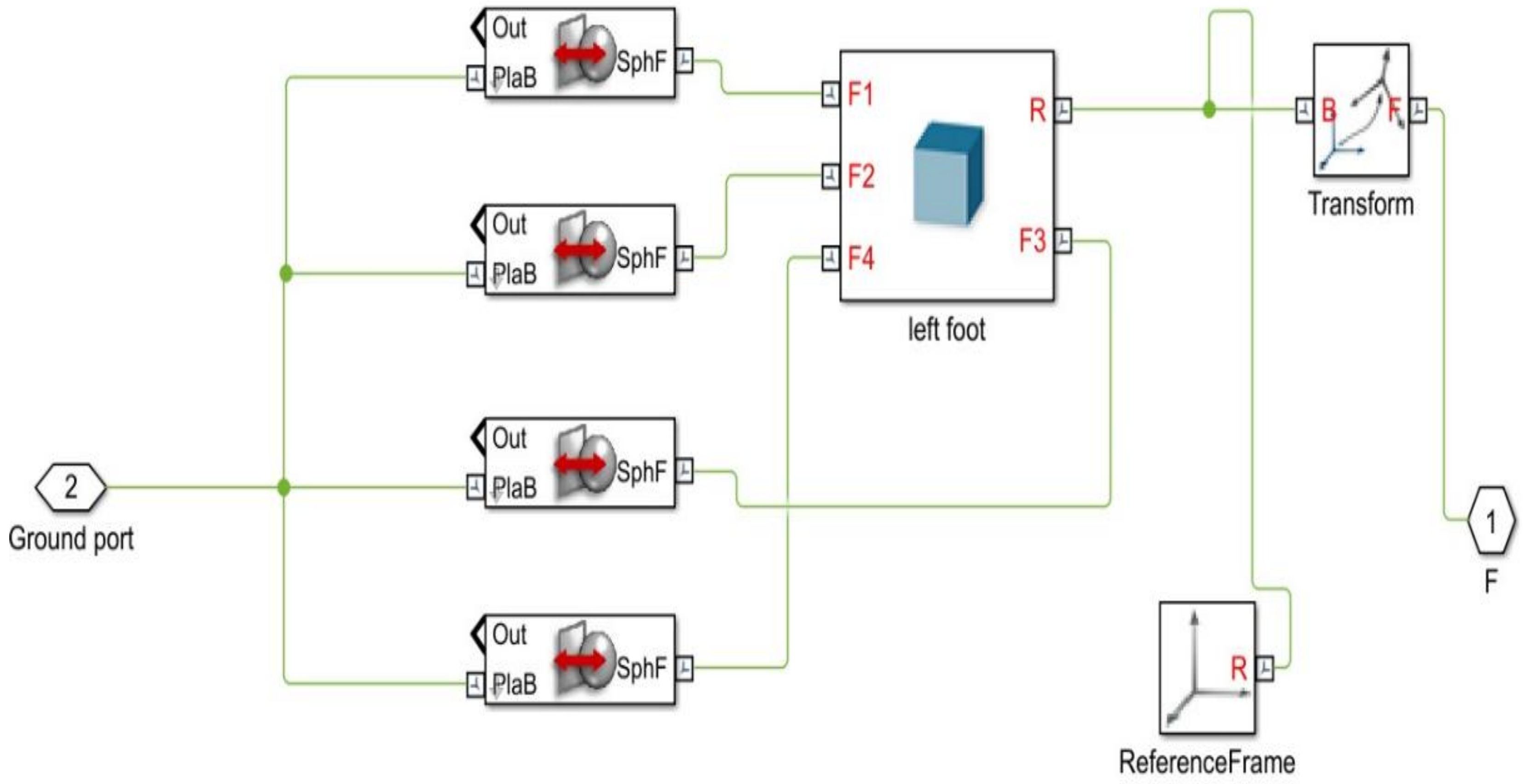


Figure 3.9 Simulink Block Diagram of Contact Forces of Left Leg of the Bipedal with the Ground

Figure 3.10 shows the Simulink model of the lower body falling of bipedal which falls when tried to walk and could not stand also.

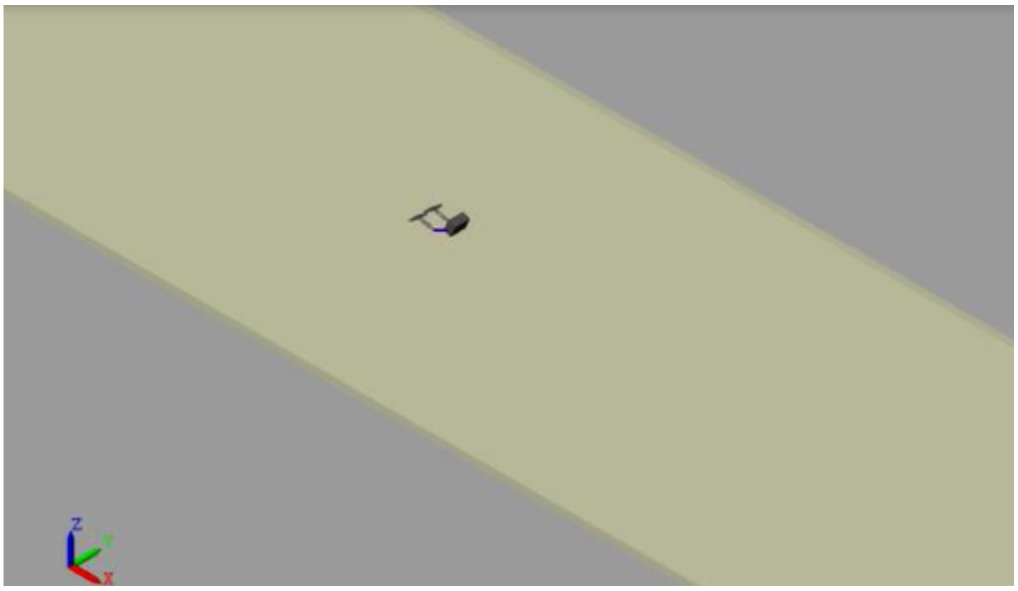


Figure 3.10 Simulink Model of Lower Body of Bipedal with Instability (Falling Down)

Figure 3.11 and Figure 3.12 shows the Simulink model of bipedal with implementation in which the bipedal prevent itself from falling.

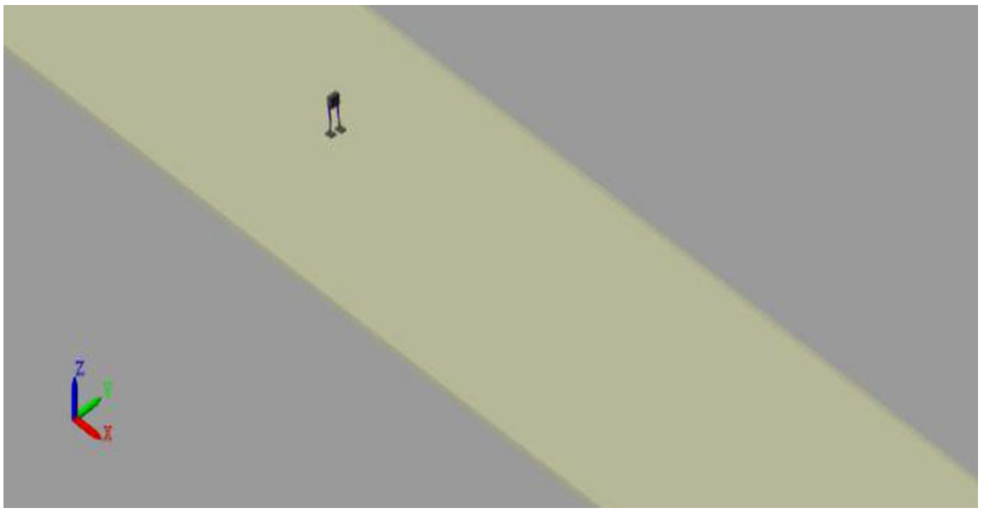


Figure 3.11 Simulink Model of Bipedal in execution with Prevention from Falling

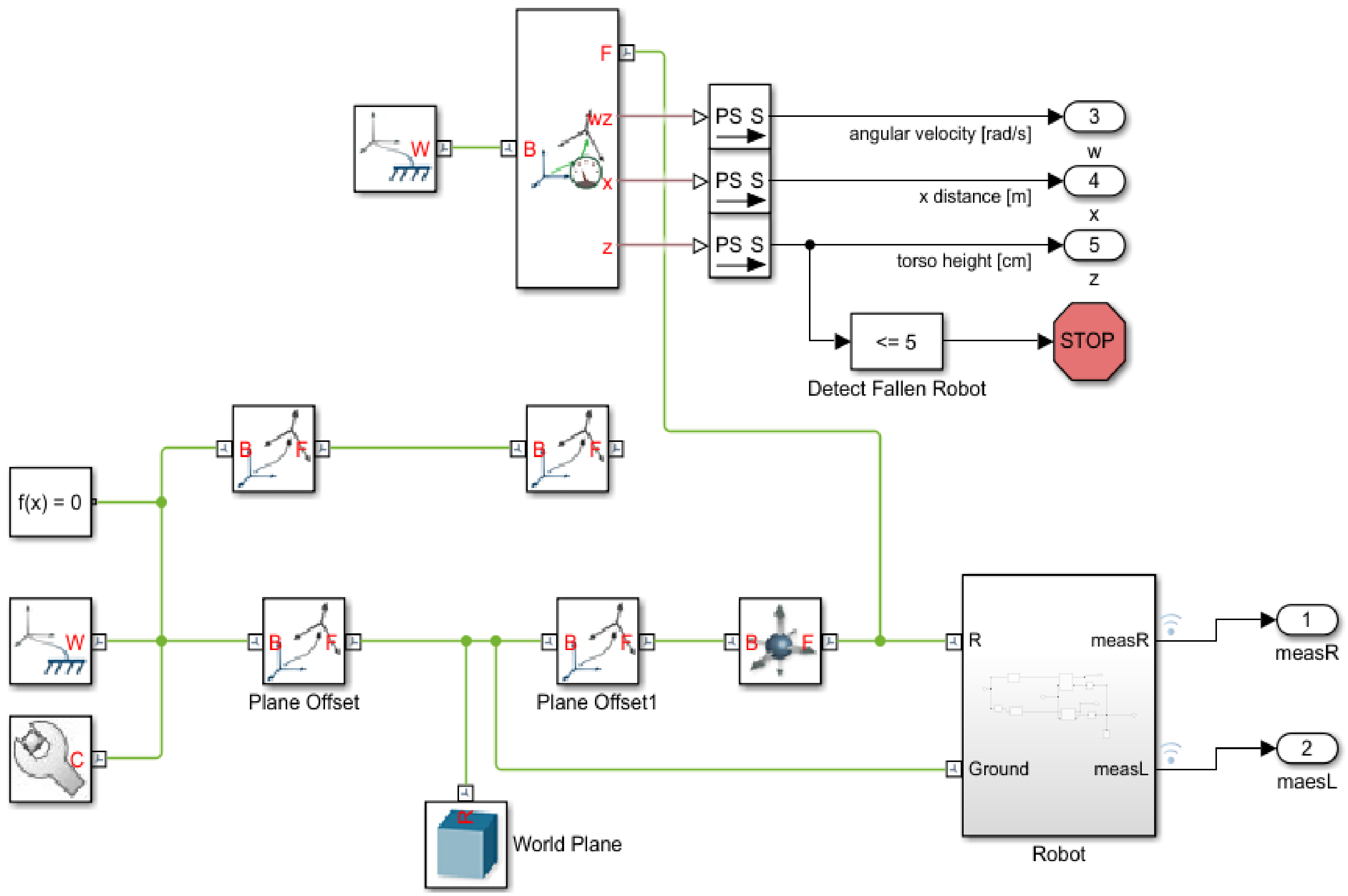


Figure 3.12 Simulink Model of Bipedal with Prevention of Falling

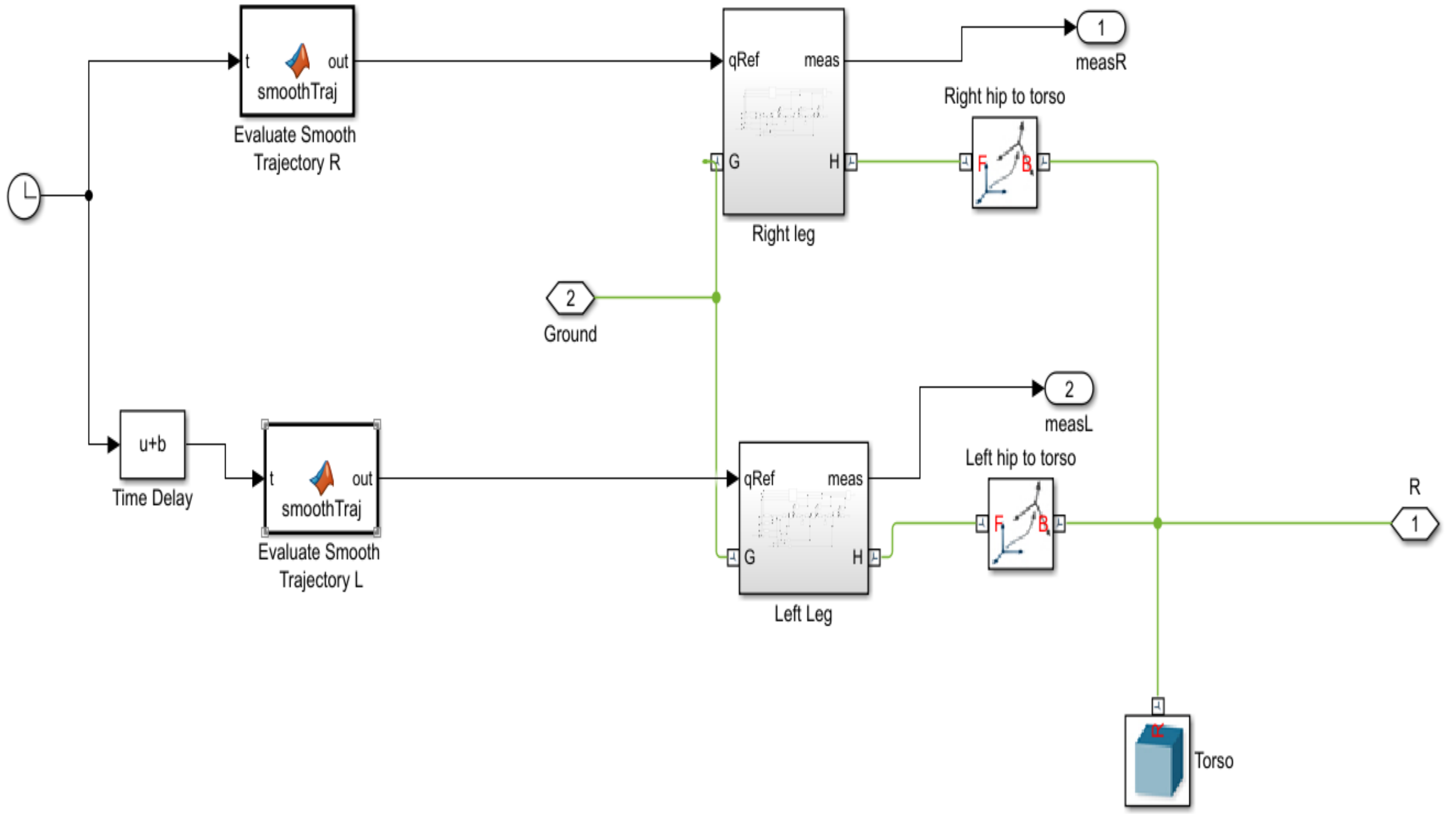


Figure 3.13 Simulink Model of Bipedal for Smooth Trajectory

Figure 3.13 shows how the bipedal trajectory can be smooth without jerks after once the proposed algorithm is executed, then thereafter the optimal state-action values are being stored as MATLAB algorithm codes are executed. Detailed Matlab codes are in Appendix D.

3.4 Simulink Model of Bipedal with the Object Identification and Localization

Figure 3.15 shows the Simulink model of bipedal with an object (in this case soccer ball) placed in a dynamic and uncertain environment.

Figure 3.16 and Figure 3.14 exhibit the Simulink model of bipedal with user define code (Matlab function) for object localization (in this case Soccer ball) in a dynamic and uncertain environment(Sharma et al., 2020).

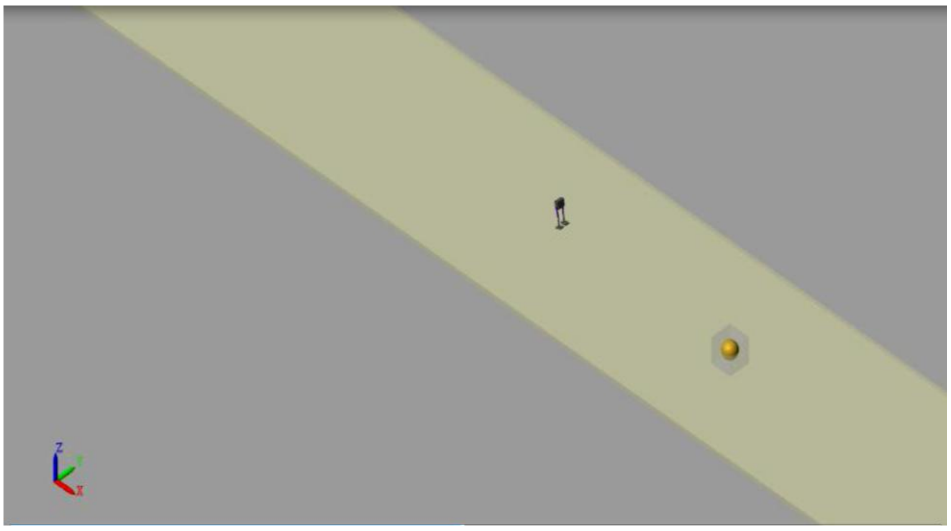


Figure 3.14 Simulink Model of Bipedal with Localization Code execution in Dynamic Environment

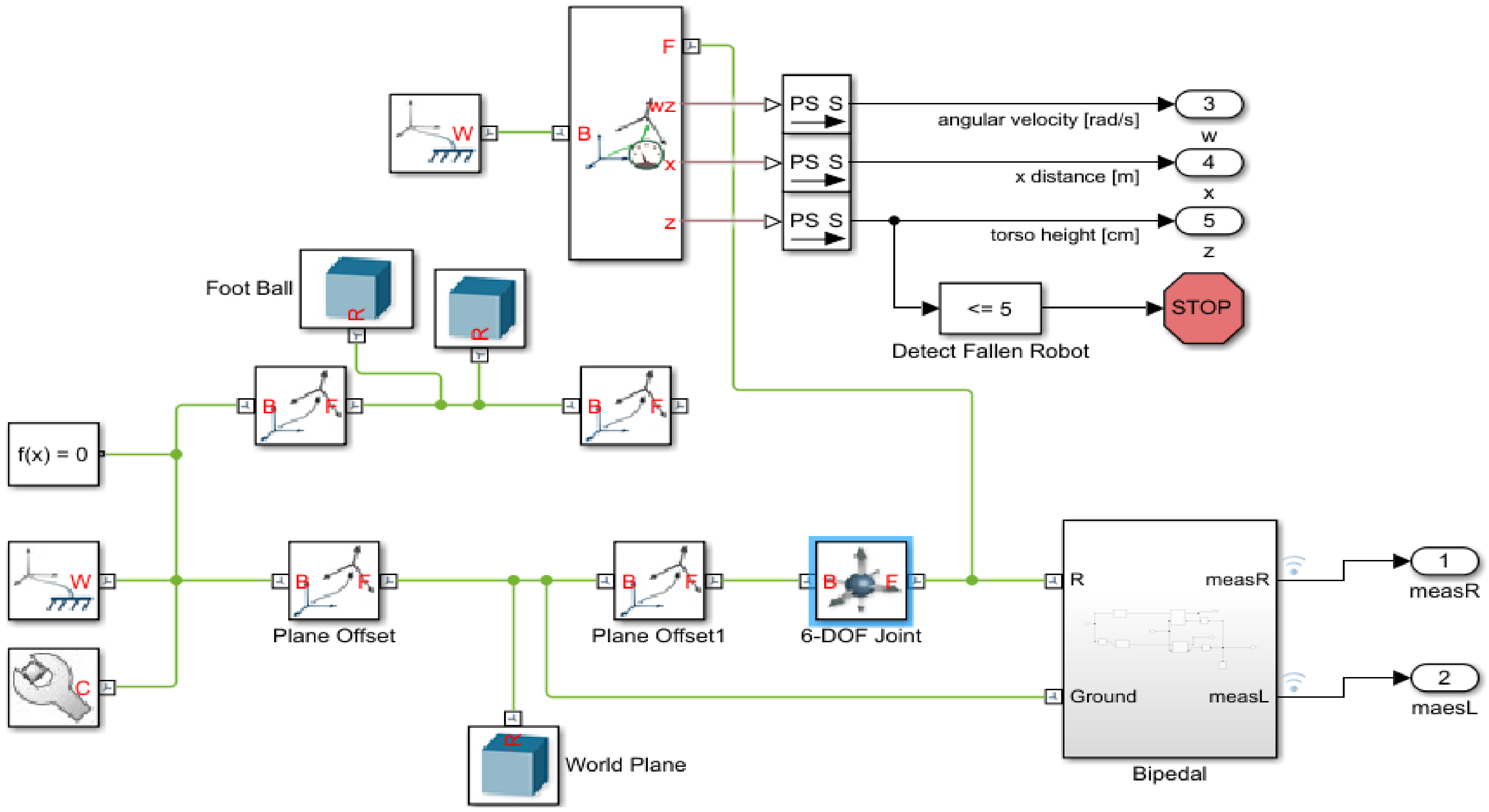


Figure 3.15 Simulink Model of Bipedal with Object in Dynamic Environment

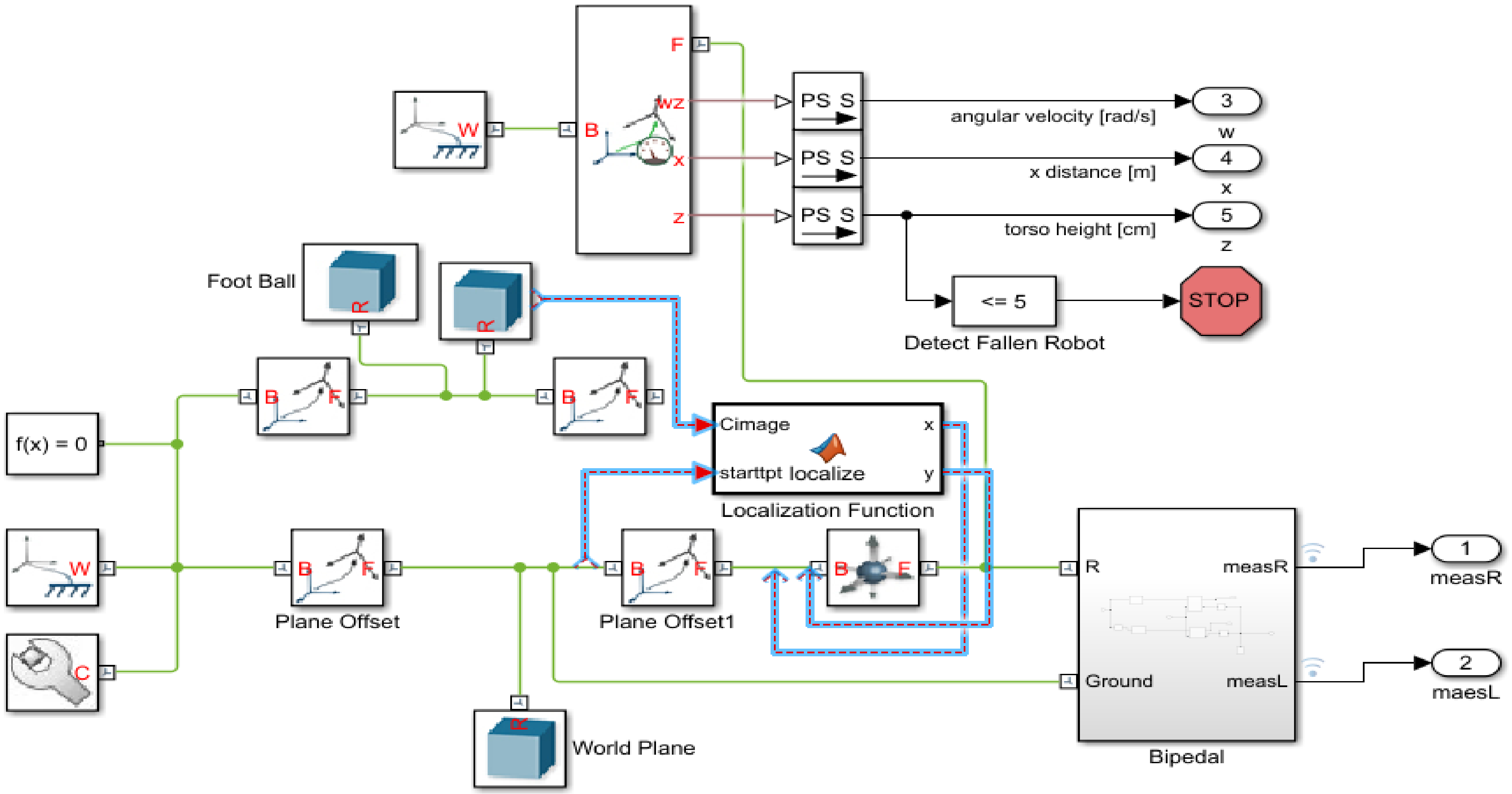


Figure 3.16 Simulink Model of Bipedal with Object Localization Code in the Dynamic Environment

CHAPTER 4 BIPEDAL WALKING ROBOT: ARCHITECTURE

4.1 Overall System is designed to achieve Sub-Objectives

Figure 4.1 shows the general framework which is intended to accomplish the desired sub-objectives.

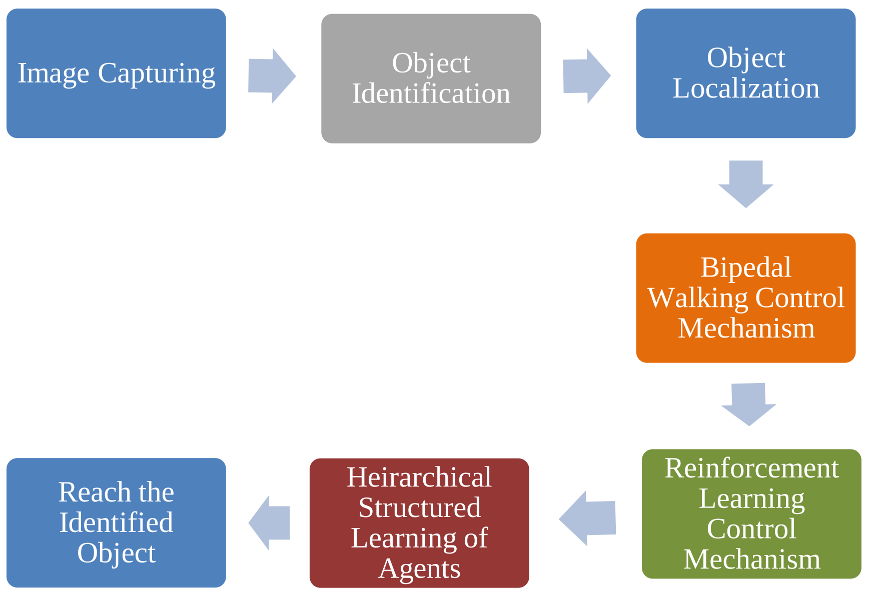


Image Capturing
Object Identification
Object Localization
Bipedal Walking Control Mechanism
Reinforcement Learning Control Mechanism
Heirarchical Structured Learning of Agents
Reach the Identified Object

Figure 4.1 Overall System Designed to achieve the Sub-Objectives

4.2 Model of the Bipedal for Object Identification and Navigation

Steps followed by the bipedal to identify the object and navigate to the identified object are:

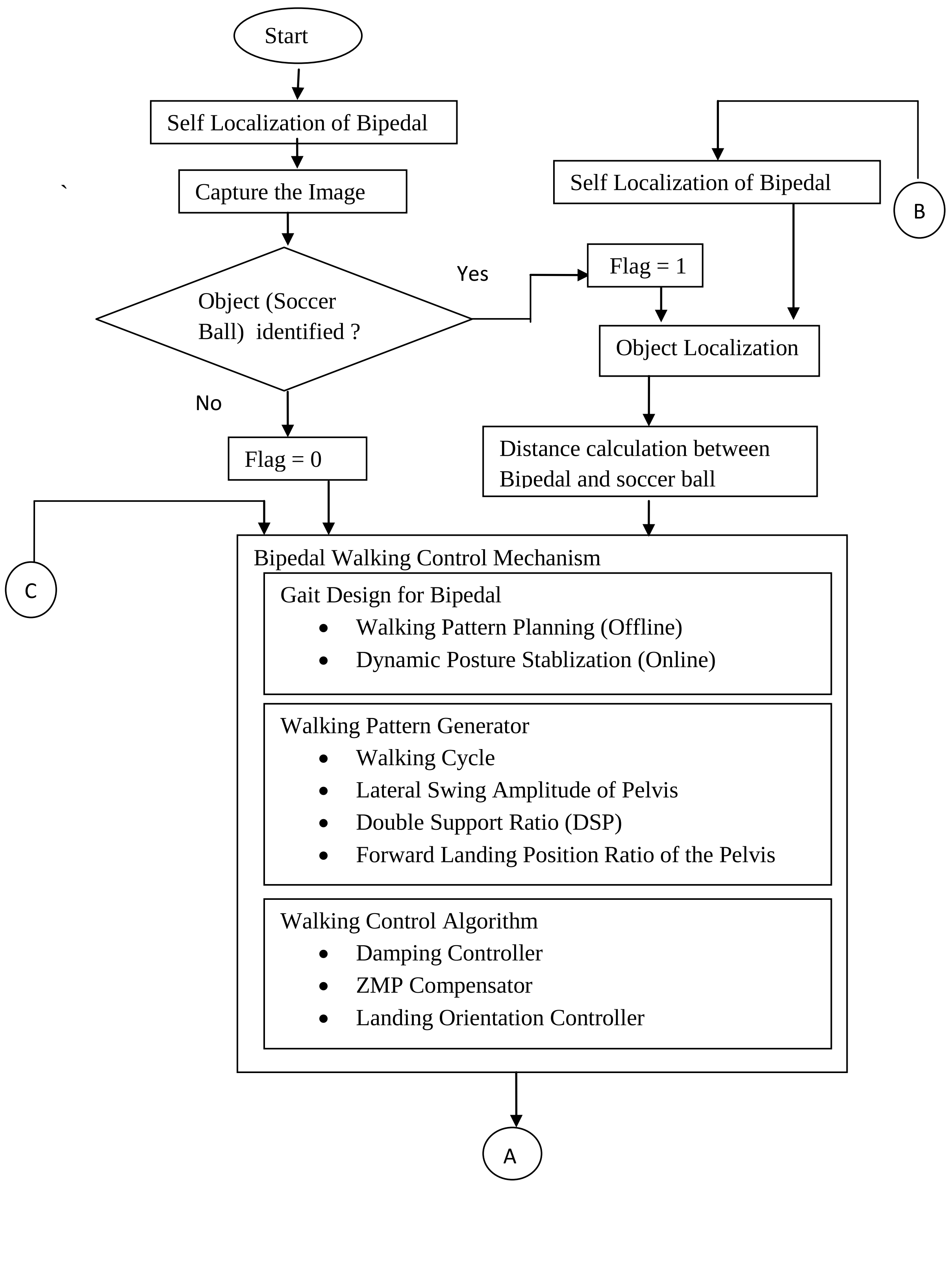
1. The object is seen by the bipedal through the vision sensor.
2. The object seen is then compared with the objects stored in the database using the SURF algorithm.
3. The localization of the identified object is carried out.
4. The bipedal now gets the location to reach the object as the goal point.

5. The bipedal now start hierarchically learning each joint i.e. first learn hip joint then knee joint and then ankle joint of each leg.
6. After learning is carried out by reinforcement learning control algorithm.
7. The control mechanism is implemented on the bipedal joints so that the bipedal stays in a stable balanced state and walks to the desired location.
8. Then by using the learned data stored in the lookup table for the same environment the bipedal walk stably without jerks.
9. For a new dynamic environment, it learns from beginning/ scratch and stores the optimal actions and policy in the lookup table. These learned and stored data can be used to execute in the future.

4.3 Flow Diagram of the Overall System

The activities of the overall system include:

1. Self Localization of the bipedal
2. Object Identification
3. Object Localization
4. Distance calculation between bipedal and the object (soccer ball)
5. Bipedal Walking Control Mechanism
    a. Gait Design for bipedal
    b. Walking Pattern Generator
    c. Walking Control Algorithm
6. Reinforcement Learning Control Mechanism
    a. Action Selection
    b. Reward Calculation
    c. Finding the Optimal Action
    d. Updation of values
7. Hierarchical Structured Learning Reinforcement learning Agents
    a. Implement control mechanism for Hip joint Trajectory
    b. Implement control mechanism for Knee joint Trajectory
    c. Implement control mechanism for Ankle joint Trajectory
    d. Contact force execution on the Foot Sole
8. Reaching Goal Position near the object


Start
Self Localization of Bipedal
Capture the Image
Object (Soccer Ball) identified ?
Yes
No
Flag = 1
Flag = 0
Self Localization of Bipedal
B
Object Localization
Distance calculation between Bipedal and soccer ball
C
Bipedal Walking Control Mechanism
Gait Design for Bipedal
Walking Pattern Planning (Offline)
Dynamic Posture Stablization (Online)
Walking Pattern Generator
Walking Cycle
Lateral Swing Amplitude of Pelvis
Double Support Ratio (DSP)
Forward Landing Position Ratio of the Pelvis
Walking Control Algorithm
Damping Controller
ZMP Compensator
Landing Orientation Controller
A

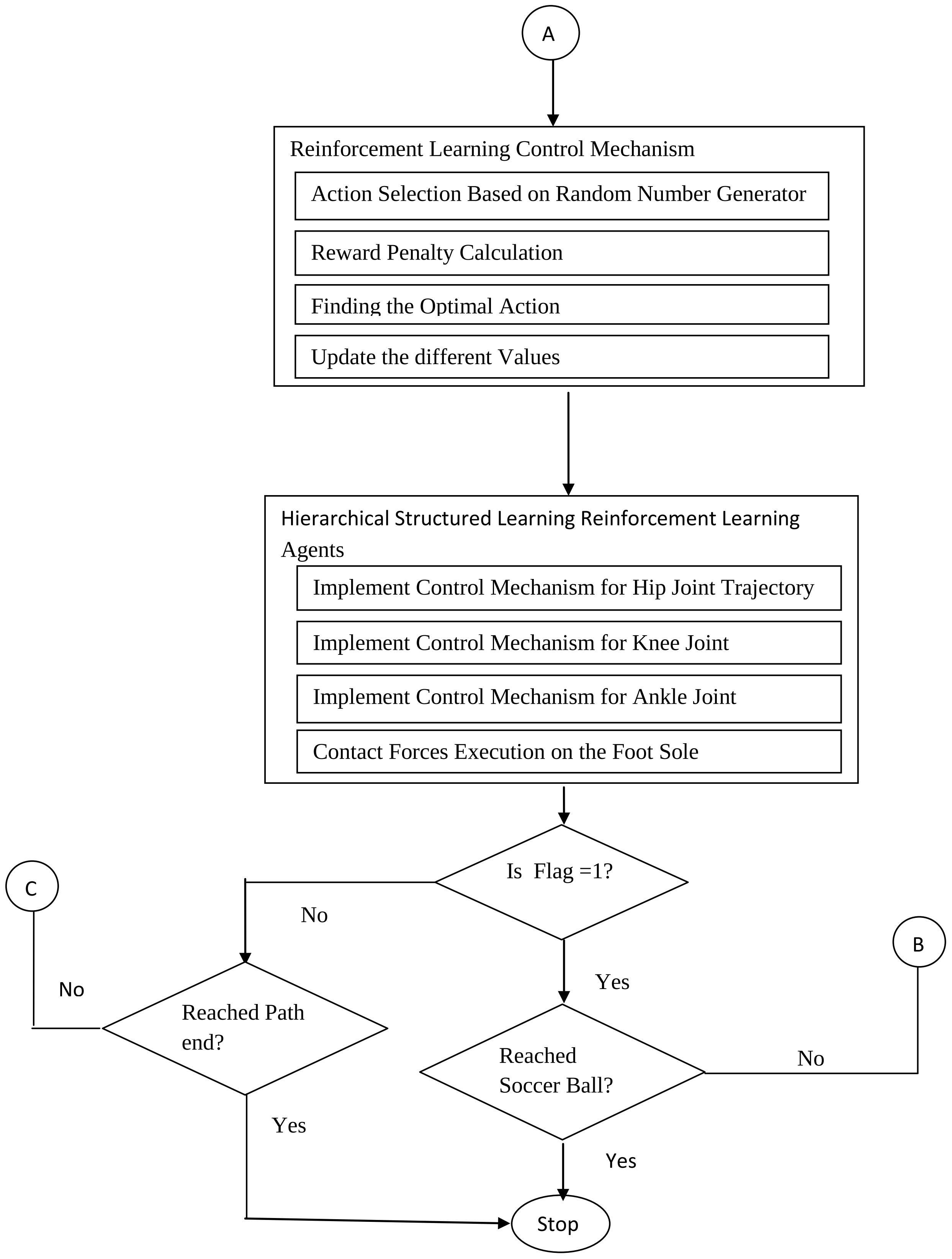

A
Reinforcement Learning Control Mechanism
Action Selection Based on Random Number Generator
Reward Penalty Calculation
Finding the Optimal Action
Update the different Values
Hierarchical Structured Learning Reinforcement Learning Agents
Implement Control Mechanism for Hip Joint Trajectory
Implement Control Mechanism for Knee Joint
Implement Control Mechanism for Ankle Joint
Contact Forces Execution on the Foot Sole
Is Flag =1?
No
Yes
C
No
Reached Path end?
Yes
Reached Soccer Ball?
No
B
Yes
Stop

Figure 4.2 Flow Chart of the Overall System

4.3.1 System Dynamics and Variations

The highly abstract method of modeling is being used here ignoring the fine details of the dynamic environment. The main work included the self localization of the bipedal, object localization in the environment, finding the distance between the bipedal and object (soccer ball) and then reaching the object using the Reinforcement learning algorithm (forgetting mechanism incorporation in traditional Q-learning) which follows the optimal policy to reach the object (in this case soccer ball). In future when more objects along with other players on the ground are being added in the dynamic environment of a soccer match scenario along with motion then work has to done in computer vision part of the bipedal.

4.4 Stepwise Execution of the Overall System

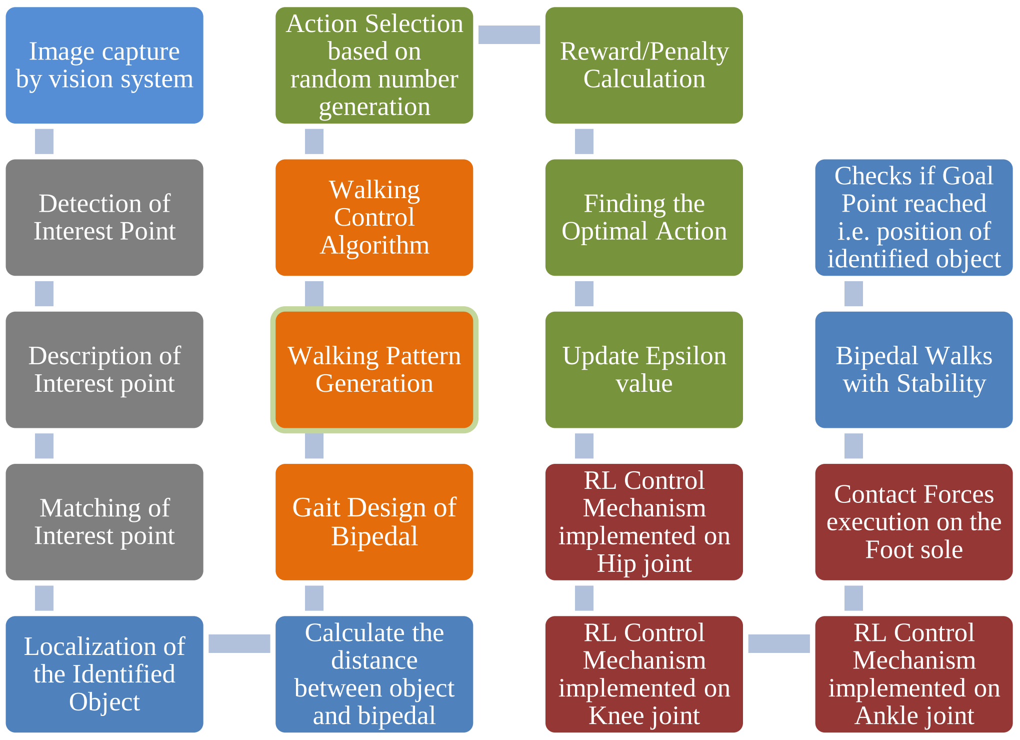


Image capture by vision system
Detection of Interest Point
Description of Interest point
Matching of Interest point
Localization of the Identified Object
Calculate the distance between object and bipedal
Gait Design of Bipedal
Walking Pattern Generation
Walking Control Algorithm
Action Selection based on random number generation
Reward/Penalty Calculation
Finding the Optimal Action
Update Epsilon value
RL Control Mechanism implemented on Hip joint
RL Control Mechanism implemented on Knee joint
RL Control Mechanism implemented on Ankle joint
Contact Forces execution on the Foot sole
Bipedal Walks with Stability
Checks if Goal Point reached i.e. position of identified object

Figure 4.3 Stepwise Execution of the Overall System

4.5 Architectural Mechanism of Object Identification

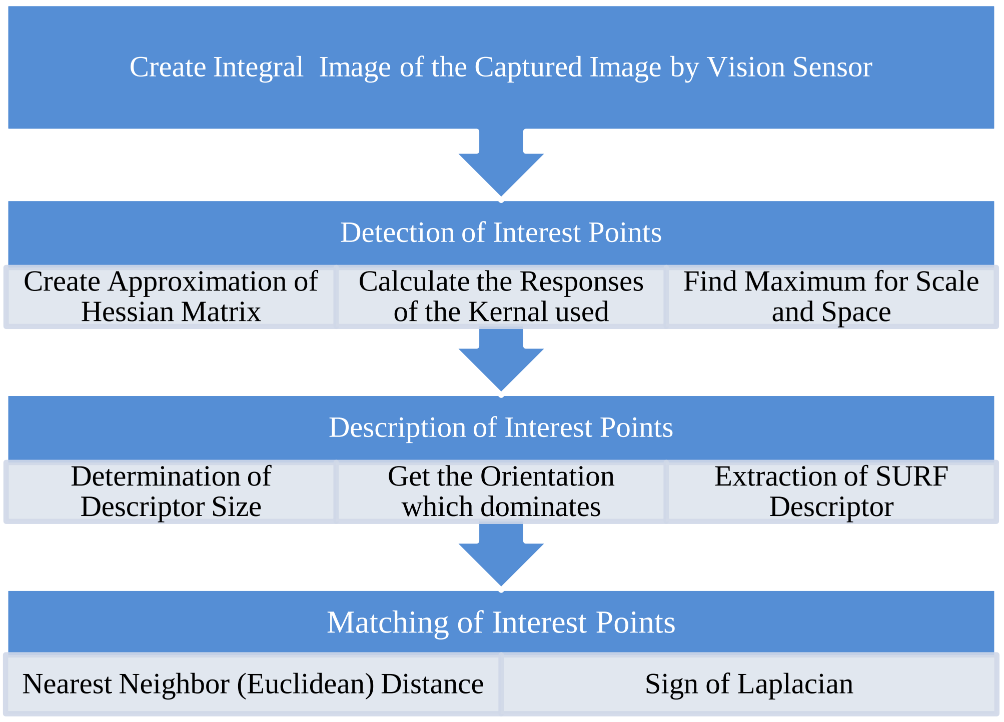

Create Integral Image of the Captured Image by Vision Sensor
Detection of Interest Points
Create Approximation of Hessian Matrix
Calculate the Responses of the Kernal used
Find Maximum for Scale and Space
Description of Interest Points
Determination of Descriptor Size
Get the Orientation which dominates
Extraction of SURF Descriptor
Matching of Interest Points
Nearest Neighbor (Euclidean) Distance
Sign of Laplacian

Figure 4.4 Object Identification Mechanism

The image is captured by the vision sensor on the torso. Then captured image is compared with a stored reference image of an object in the database.

4.6 Mechanism for Localization of the Object

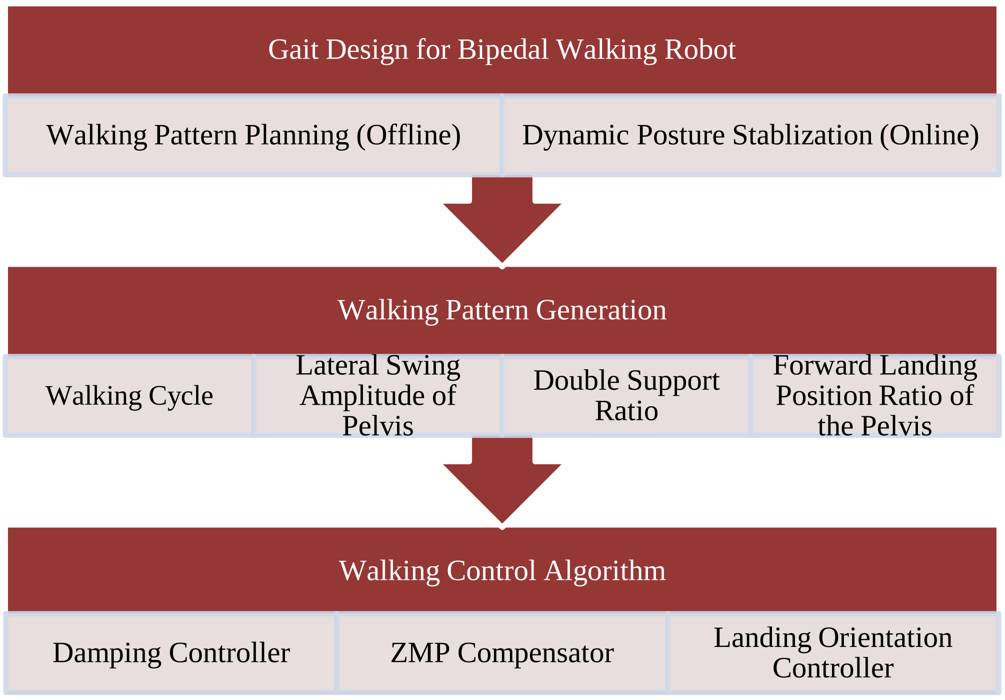
1. The coordinates are evaluated of the captured image of the object matched on the frame.
2. The bipedal calculates its current position and knows the exact coordinates of an object identified in a dynamic environment.
3. Bipedal calculates the actual distance by finding the difference between the current coordinate location and coordinates of the bottom left corner of the identified object image.

Now, bipedal has to navigate to the identified object using a controlled reinforcement learning mechanism. While navigating the bipedal joints have to be trained by the proposed algorithm.

4.7 Architectural Model of Control Mechanism of Bipedal

The steps followed for Control Mechanism designing of bipedal includes :

1. Gait Design for Bipedal Walking Robot
2. Walking Pattern Generation
3. Walking Control Algorithm



Gait Design for Bipedal Walking Robot
Walking Pattern Planning (Offline)
Dynamic Posture Stablization (Online)
Walking Pattern Generation
Walking Cycle
Lateral Swing Amplitude of Pelvis
Double Support Ratio
Forward Landing Position Ratio of the Pelvis
Walking Control Algorithm
Damping Controller
ZMP Compensator
Landing Orientation Controller

Figure 4.5 Control Mechanism for Gait of the Bipedal

Usually, an industrial robot has a fixed base. But in the bipedal robot base is not fixed. The bipedal walking robot moves around with difficulty and may be critical in dynamic conditions. The motion of the bipedal walking robot maintains contact between the sole and the ground. In standing condition, the weight of the body is vertically down and the reaction force acts in a vertically up direction and there is no horizontal component of a force acting. Hence, the ZMP is not disturbing, and bipedal is maintaining the static balance.

Gait trajectory is structured offline to make a robot walk. In bipedal mechanical autonomy look into the field, the gait trajectory produces the relative position directions of both the feet concerning the pelvis center. Due to large upper body motions, the robot will fall even after designing a fullproof walking pattern (Figure 4.6). Due to this the upper body including shoulders and hands are not taken into consideration in the proposed work.

Zero moment point (ZMP) inspects static and dynamic forces. Numerous analysts and researchers had suggested techniques dependent on the ZMP criterion for stable walking (Sutton, 1990). If the control system designed is capable to keep the position of ZMP within scope or polygon formed of the soles. This will help the robot to walk steadily (Figure 4.6).

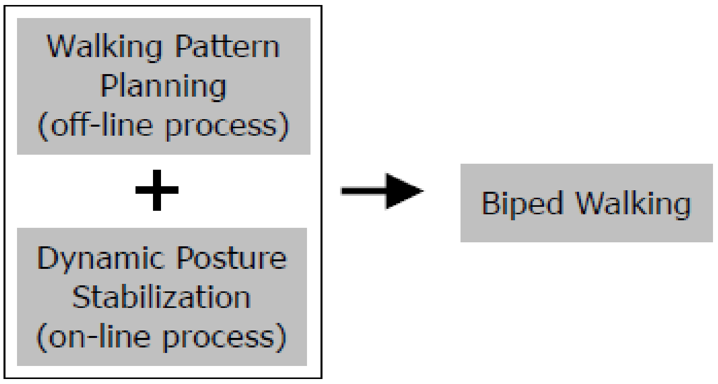

Walking Pattern Planning (off-line process)
+
Dynamic Posture Stabilization (on-line process)
Biped Walking

Figure 4.6 Realization of the Bipedal Walking

To enhance the robustness of bipedal robots walking learning methods on bipedal walking have been studied{(Ogura et al., 2006) (Akachi et al., 2005) (Ha et al., 2011)(Endo et al., 2008)}. The RL agent collects the training experiences, which act as experiences for the next coming training set, and through interaction in the dynamic environment, the learning policy is updated. The trial and error method is used for the learning process which helps in obtaining the walking policy instead of using past training experience in advance. The feedback positions from ZMP are analyzed to find whether the robot is in a stable state or not and to avoid falling/tipping of bipedal.

The bipedal ought to keep inside a support polygon, which characterizes a convex hull formed by all contact focuses on the floor. After learning for the long haul the framework gets a walking policy that fits the current dynamic and consistently evolving conditions.

4.8 Architectural Model of Reinforcement Learning Control Mechanism

4.8.1 Reinforcement Learning Control Mechanism

In a dynamic environment, for bipedal to navigate the following constraints are considered:

- States are consistently dispersed between Start state (current position of Bipedal) and Goal state (known set) (position calculated by localization of object)
- The feasible set of actions A= {0,1} (known set). For certain cases likewise A= {-1,0,1}
- α (Learning rate) is considered 0.9
- λ (Discount Factor) is considered 0.9
- ε (Exploration probability) is taken 0.5 (1- ε is for exploitation)
- ε-decay is considered 0.98. This is also known as the forgetting factor which is proposed in current work.

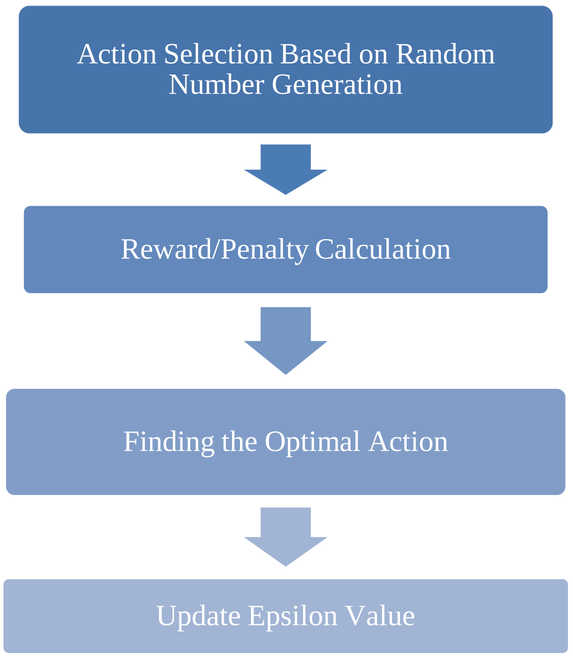

Action Selection Based on Random Number Generation
Reward/Penalty Calculation
Finding the Optimal Action
Update Epsilon Value

Figure 4.7 Reinforcement Learning Control Mechanism of the Bipedal

- A tradeoff between Exploitation and Exploration is observed - for exploitation the RL agent responds slowly to the evolving environment, for exploitation the RL agent quickly adapts to the consistently dynamic environment. The proposed framework is empowering investigation/ exploration as it is progressively compelling in managing the consistently dynamic conditions.

4.8.2 Hierarchical Structured Learning of RL Agents

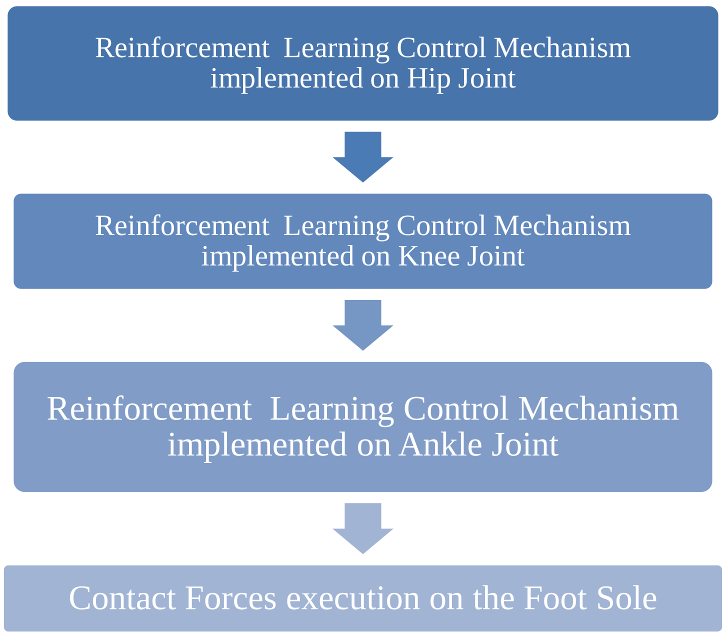

Reinforcement Learning Control Mechanism implemented on Hip Joint
Reinforcement Learning Control Mechanism implemented on Knee Joint
Reinforcement Learning Control Mechanism implemented on Ankle Joint
Contact Forces execution on the Foot Sole

Figure 4.8 Hierarchical Structured Learning of RL Agents

The hierarchical learning of RL agents takes places as shown in Figure 4.8 first the hip joint which is attached to the pelvis is learned, then learning of the knee joint to maintain the stability of the bipedal, then learning of the ankle joint is done taking into consideration of damping when the sole is in contact with the ground.

CHAPTER 5 BIPEDAL WALKING ROBOT: MATHEMATICAL MODEL, CONTROL

5.1 Trajectory of Bipedal

The basics for the trajectory of the bipedal is Biomechanics. Biomechanics is the field of science that applies the laws of mechanics and physics to the movement and the structure of all living organisms and their performance. This field in the special case also deals in the force exerted by muscles and gravity on the skeletal structure of humans. Biomechanics suggests that a humanoid should form a closed polygon when it is in motion. The best design for mechanical stability is a closed-loop polygon. A trajectory is the sequence of movement of the individual joint. (Figure 5.1)

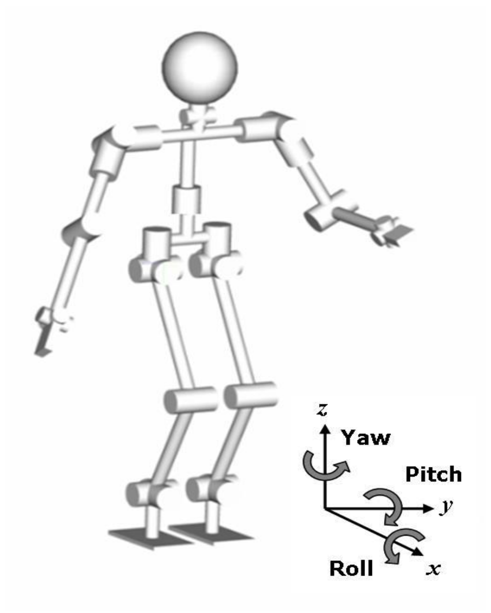


Figure 5.1 Bipedal Walking Robot

The proposed bipedal model has ten degrees of freedom. Every leg has five degrees of freedom, the torso of bipedal, three degrees of freedom provided for the fast movement, and to prevent from sudden falling, the knee has one DOF and ankle has one DOF. Design principle of development of the bipedal resembling the human body. In the proposed model, the assumption is that counter generated by the left leg is identical to the counter generated by the right leg. The upper body weight acts on both feet and which supports the balancing of the body. The bipedal body is symmetric about the sagittal plane.

5.2 Mathematical Model of Object Identification

SURF interest points are in-plane rotation-invariant, robust to noise, and overall, extremely fast to calculate. The three steps followed are:

1. Identification of Interest Point
2. Depiction of Interest Point
3. Matching of Interest Point

5.2.1 Identification of Interest Point

Interest point is the points at a specific location that are chosen. The selected locations in the image are distinct and can be corners, blobs, T-junction. Detectors should be repeatable, which helps in getting the same interest points (physically) in different viewing conditions.

5.2.1.1 Integral Images

Integral images are an image whose each pixel is the cumulative sum of all well-defined space of all pixels of input image I. Sum of areas is represented by $I_\Sigma(X)$ for location $X=(x, y)^T$. Areas are usually bounded by origin (0,0) and X.

$$I_\Sigma(X) = \sum_{i=0}^{i\le x} \sum_{j=0}^{j\le y} I(i,j) \tag{5.1}$$

Integral images are incredibly efficient. It is possible to characterize a region of the image using three operations and four memory accesses.

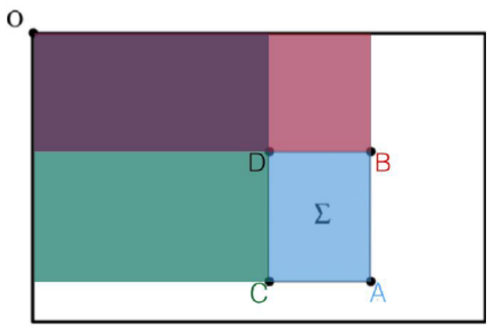


Figure 5.2 Basics of Integral Image

$$\Sigma = A - B - C + D \qquad (5.2)$$

5.2.1.2 Hessian-Based Interest Points

In blob detection, part of images is detected which differs in properties- color, brightness, surrounding regions, and so on. This detector detects blob-like structures using the determinants where the Hessian is maximum. This gives good performance accuracy.

Due to discretization repeatability is maximum at multiples of π/2 due to the square format of filters while at odd multiples of π/4 some repeatability is lost. The detailed description is in Appendix E.

5.2.1.3 Hessian Approximation

The actual calculation of the Hessian matrix is slow. Instead, Hessian can be approximated using Box filters. The relative weights *w* are taken simply for computational efficiency and balances the Hessian's determinant expression.

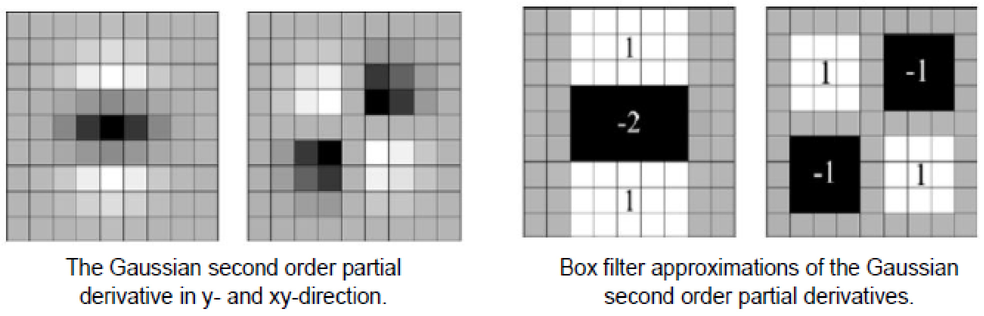

1
-2
1
1
-1
-1
1
The Gaussian second order partial derivative in y- and xy-direction.
Box filter approximations of the Gaussian second order partial derivatives.

Figure 5.3 Interest Point Detection using Discretized and Cropped Gaussian (in the first part), Box Filter Approximation (in the second part)

$$\det(H_{approx}) = D_{xx}D_{yy} - (w\,D_{xy})^2 \qquad (5.3)$$

Where D_{xx} is an approximation of 2nd order Gaussian partial derivative in the X-direction, w is taken 0.9.

5.2.1.4 Representation of Scale Space

To coordinate interest points across various scales, a pyramidal scale space is created. Instead of serial downsampling, each progressive degree of the pyramid is developed by scaling up in parallel. This has the advantage of computational efficiency.

For each new octave, filter size doubles (6-12 to 24-48) resulting in a sampling interval for interest point extraction to be doubled which reduces computation time. The detailed description is in Appendix E.

5.2.1.5 Localization of Interest Point

Localization of interest points is done by suppression of non-maximum (non-maximum pixel are set to 0) points in the neighborhood of 3x3x3. Interpolation in terms of scale and image space is done for the maximum value of the Hessian matrix grid determinant. The detailed description is in Appendix E.

5.2.2 Description of Interest Point

The interest point neighborhood detection for blob response uses 1st order Haar wavelet reaction in x and y directions. The detailed description is in Appendix E

5.2.2.1 Orientation Assignment

Orientation assignment reduces the time duration of feature computation and feature matching which increases robustness. The Haar wavelet responses are calculated in both directions in a neighborhood defined by a circle within the radius of 6s. Interest point centers are weighted with Gaussian taking σ = 2s and plots of directional strengths are made.

These plots are divided into sliding orientation windows and local orientation vectors are computed as the sum of x and y responses within each window. The dominant orientation is the largest of all such vectors across all windows.

5.2.2.2 Feature Vector

To extract features, an axis orientated 20s sized square window is defined, a window is subdivided into a 4x4 grid. The horizontal and vertical Haar wavelet response is calculated over each subdivision and four metrics are extracted from each subdivision using 5x5 equally spaced points. These metrics are then summed to produce the local feature vector which is concatenated to form a 64-element feature vector that describes the interest point and surrounding neighborhood.

$$v = \begin{bmatrix} \sum d_x \\ \sum d_y \\ \sum |d_x| \\ \sum |d_y| \end{bmatrix} \tag{5.4}$$

where d_x - the reaction of Haar Wavelet in horizontal axis

d_{y}- the reaction of Haar Wavelet in the vertical axis.

5.2.3 Matching Interest Points

5.2.3.1 Nearest Neighbor

Features are matched across frames as the nearest neighbor within a distinct feature threshold. Either Euclidean or Mahalanobis distance may be used to determine 'nearest'. In this implementation, uniform precision was assumed and therefore, Euclidean distance was sufficient.

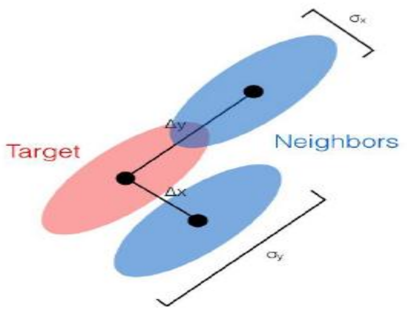


Figure 5.4 Euclidean Distance

$$D_E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (5.5)$$

$$D_M = \sqrt{\frac{(x_2 - x_1)^2}{\sigma_x^2} + \frac{(y_2 - y_1)^2}{\sigma_y^2}} \qquad (5.6)$$

5.2.3.2 Laplacian Indexing

In the matching phase, the Laplacian sign (Tr(H)) is utilized for fast indexing. Discrimination cascade includes sign. The sign of Laplacian helps in distinguishing bright/ white blobs on dark/ black backgrounds from the opposite situation and serves as a meaningful metric to divide the set of all interest points.

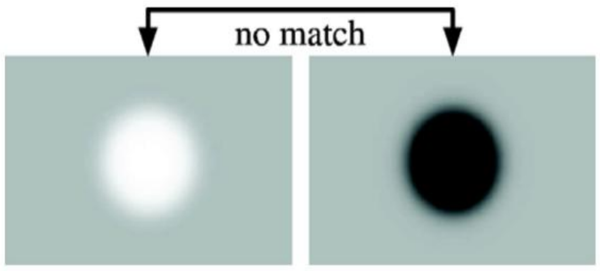


Figure 5.5 Sign of Laplacian

5.3 Mathematical Model of Localization of Object

1. The coordinates are evaluated of the object matched on the frame.
2. The bipedal calculates its current position and knows the position of the object identified.
3. Bipedal finds the distance by finding the difference between its current location and the bottom left corner of the object identified.

Now the bipedal has to navigate to the identified object using a controlled reinforcement learning mechanism.

5.4 Mathematical Model of Control Mechanism of Bipedal

The steps followed for control mechanism designing of bipedal includes(Sharma et al., 2020) :

1. Gait Design for Bipedal Walking Robot
2. Generating Walking pattern
3. Walking Control Algorithm

5.4.1 Generating Walking Pattern

Consider the following four factors for designing the walking pattern of bipedal (Sharma et al., 2020; Sharma, Singh, Bharadwaj, et al., 2019):

1. Walking cycle (twice of step time)
2. Lateral swing amplitude of pelvis
3. Double support ratio (DSR)
4. Forward landing position ratio of the pelvis

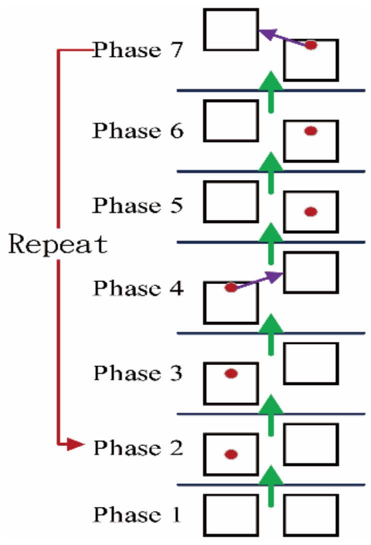


Figure 5.6 ZMP Position of a Biped Walking Sequence

5.4.1.1 Walking Cycle

The walking cycle is set as a regular recurrence of a 2D simple inverted pendulum model (Figure 5.7). Assuming, frequency of inverted pendulum f_n,

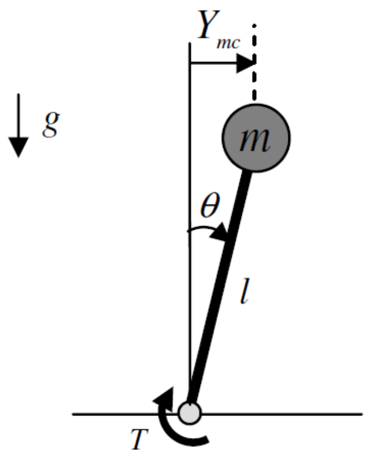


Figure 5.7 Inverted Pendulum Model

$$f_n = \frac{1}{2\pi}\sqrt{\frac{g}{l}} \tag{5.7}$$

where *l* - pendulum length.

5.4.1.2 Lateral/Sideway Swing Amplitude of Pelvis

The sideway swing extent of the pelvis is obtained by the ZMP fluctuation of IPM. Motion equation of IPM is given as:

$$T = mgl - ml^2\ddot{\theta} \tag{5.8}$$

Where T - joint torque, m - point mass and θ- angular displacement.

Divide both sides of equation 5.8 by *mg*, then the equation becomes

$$\frac{T}{mg} = l\Theta - \frac{l}{g}(l\ddot{\Theta}) \tag{5.9}$$

By substituting $F_z = mg$ and $Y_{mc} = l\theta$ into Equation 5.9 assuming θ is very small (< 5°), then the equation becomes

$$\frac{T}{Fz} = Y_{mc} - \frac{l}{g}(\ddot{Y_{mc}}) \tag{5.10}$$

Where F_z - ground response force, Y_{mc} - lateral displacement of mass center.

ZMP dynamics is obtained when F_z ground response force is divided by the torque T by:

$$Y_{zmp} = Y_{mc} - \frac{l}{g}\ddot{Y_{mc}} \tag{5.11}$$

Where Y_{zmp} - lateral ZMP.

While the bipedal is walking, lateral displacement is assumed as $Y_{mc}=A \sin \omega t$ (coronal plane), the equation becomes

$$Y_{zmp} = A\left(1 + \frac{l}{g}\omega^2\right)\sin\omega t \tag{5.12}$$

In real bipedal walking, as force/torque sensors are attached at ankle joints, deflection of the compliant results in an increase in amplitude *A* than the original value.

5.4.1.3 Double Support Ratio (DSR)

During a walking cycle, a double support ratio is given by the ratio of time when two feet are on the ground in contact with the floor(Sharma et al., 2020) (Figure 5.8). For humans, this ratio is about over 10 percent (Iwata & Sugano, 2009).

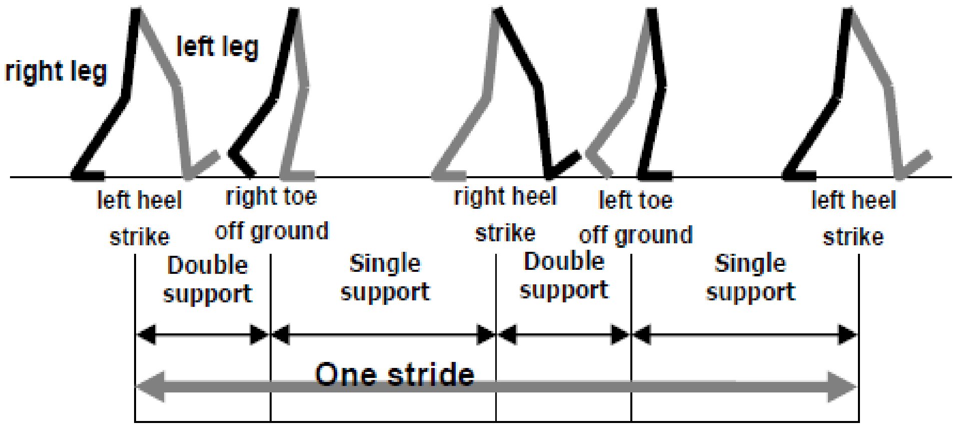


Figure 5.8 Walking Cycle

5.4.1.4 Forward Landing Position

Forward landing position ratio of pelvis γ_{pelvis} is the proportion of the position of the front leg to the rear/back leg when the double support phase begins (Figure 5.9). That is, in this situation ***if the*** proportion ***of pelvis is close to 1.0, then the front leg is closer to the pelvis at the beginning of DSP*** (Figure 5.10). The bipedal acts like an inverted pendulum swinging in the forward direction.

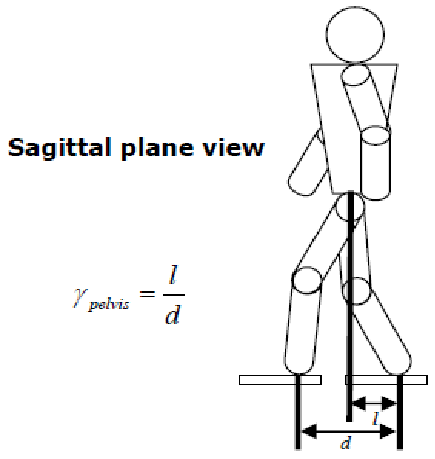


Figure 5.9 Sagittal Plane View

Assuming forward displacement as $X_{mc}=A \sin \omega t$ ZMP kinematics becomes (in the sagittal plane),:

$$X_{zmp} = X_{mc} - \frac{l}{g}\ddot{X}_{mc} \tag{5.13}$$

Where X_{mc} - forward displacement of the mass center, X_{zmp}- forward ZMP.

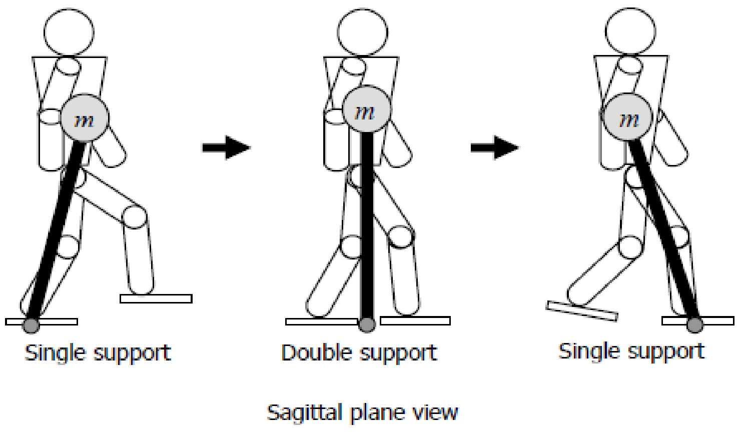


Figure 5.10 Forward Landing Position Ratio of Pelvis

In DSP, phases of X_{mc} and $\ddot{X}_{mc}$ is equal. X_{mc} is located at zero i.e. at the center position of swing trajectory and so $\ddot{X}_{mc}$ is also nearly zero.

X_{zmp} is the projected position on the ground if the pelvis is at a certain point which is at the center position between both feet.

5.4.2 Walking Control Algorithm

The walking control algorithm is based on a swapping controller. The walking cycle is separated into a few phases of walking. Appropriate controllers and their parameters are initiated during each phase/stage. Figure 5.11 depicts different stages of walking:

Stage 1 involves lifting the left leg to its maximal bending and elevation.

Stage 2 involves lowering the left leg unless entire contact with the ground is made.

Stage 3 involves lifting the right leg to its maximal bending and elevation.

Stage 4 involves lowering the right leg unless entire contact with the ground is made.

Stage 5 involves following the 1st or 3rd stage, bring the bipedal to a stable standing position when the left and right legs are completely in contact with the ground.

For stable gait, walking stages 1 to 4 are repeated consistently (Figure 5.11), so that the bipedal does not fall. In walking stages 2 and 4, a single support phase (SSP) and the double support phase (DSP) coexist.

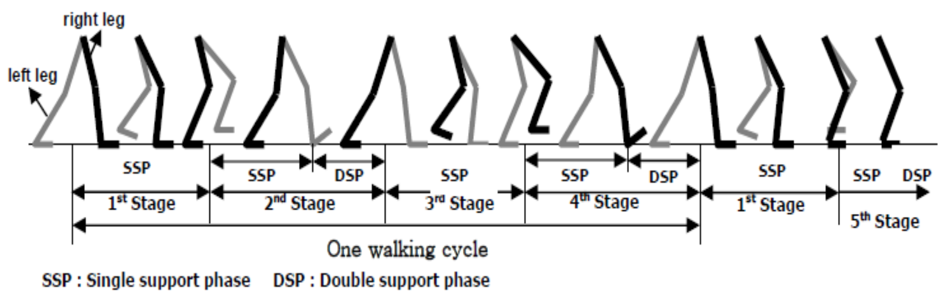


Figure 5.11 Walking Stage

A walking control algorithm comprises of three control policies(Bellemare et al., 2017; Zambaldi et al., 2018),

- Control policy for balancing (real-time),
- Control policy for walking pattern

- Control policy for predicted motion.

State #	0	1	2	3
Left Foot	Swing	Swing Contact	Stance	Heel-off
Right Foot	Stance	Heel-off	Swing	Swing Contact

- .

- **Figure 5.12 Different Stages of Left and Right Legs**

Each control policy has few controller parameters which are utilized relying on the goal required.

The force/ torque sensor at the ankle results in sustained oscillations in SSP which is overcome by damping oscillator parameters. Bipedal is a model of IP with a compliant joint.

Equation of motion is given by:

$$T = mgl\Theta - ml^2\ddot{\Theta} = K(\Theta - u) \qquad (5.14)$$

where u - reference joint angle, θ - actual joint angle due to compliance

Damping control law states

$$u_c = u - k_d\,\hat{\dot{\Theta}} \qquad (5.15)$$

where u - reference joint angle, k_d - damping control gain, and u_c - joint angle compensation.

According to ZMP dynamics, **ZMP compensator** parameters stabilize ZMP. Torso (middle body) moves back and forth and side by side. Both torso movement and ZMP are controlled by the following equation

$$Y_{zmp} = Y_{pelvis} - \frac{l}{g}\ddot{Y}_{pelvis} \qquad (5.16)$$

Where Y_{pelvis} - lateral displacement of pelvis and Y_{ZMP} - lateral ZMP.

The **landing orientation controller**, for comfortable landing, coordinates torque estimated after some time and stable contact by adjusting ankle joints to the ground.

Landing orientation control law is given as follows:

$$u_c = u + \frac{T(s)}{C_L\, s + K_L} \tag{5.17}$$

where C_L - damping coefficient, K_L - stiffness, u - reference angle of the ankle, and u_c - reference ankle angle (compensated).

In Table 5.1, bipedal walks stably using a walking control algorithm on the normal floor(Isbell et al., 2001)(Singh et al., 2005)(Sutton & Barto, 2012).

The landing timing controller helps in achieving a stable walking gait of the bipedal by updating the walking pattern schedule during landing. This prevents the biped from falling and walking unstably in the dynamic environment. The time scheduler pauses the motion if the foot does not land on the ground, the bipedal sole is not in contact with the ground.

Table 5.1 Summary of Walking Algorithm

Control Parameters	**Real-Time Parameters**	**Aim fulfilled**
Balance Controller	Damping parameters (Stages 1, 3, SSPs of 2, 4)	Reducing oscillations in the upper body in SSP (ankle joints are imposed by damping)
	ZMP compensator parameters (Stages 1,3, SSPs of 2, 4)	Maintaining balance dynamically by horizontal movement of the pelvis
Walking Pattern Control	Pelvis swing amplitude controller (Stages DSPs of 2, 4)	The amplitude of ZMP is considered to compensate lateral swing amplitude of the pelvis
Motion Control	Landing position parameters (Stages 2, 4)	Compensate landing position to prevent unstable landing

5.5 Mathematical Model of Reinforcement Learning Control Mechanism

5.5.1 Reinforcement Learning Control Mechanism

Randomness incorporated by following steps

- Generation of Random number

if random number < ***0.5*** then ***explore*** (selection of new action is done)

if random number > ***0.5*** then ***exploit*** (selection of greedy action)

- Reward/ Penalty given to RL agent builds upon variations in positions of current state and goal state

  $reward = e^{-\alpha(Goal\ state\ of\ RL\ agent - current\ state\ of\ RL\ agent)}$ (5.18)

  α - learning rate is 0.5

- The learning is continued till the value of epsilon is less than 0.001 the value of epsilon is updated in each epoch by

  $\text{epsilon} = epsilon * \text{epsilon} - \text{decay}$ (5.19)

  ε =0.5 ε-decay = 0.98

  which incorporates the forgetting mechanism in the bipedal.

Q-learning Algorithm after incorporating the randomness

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \lambda \max_{a'} Q(s', a') - Q(s,a)]$ (5.20)

r - immediate reward calculated on the fly

α - learning rate (0.5)

λ - discounted factor (0.9)

Q(s,a) - current state s when action a is taken

Q(s', a') - by taking action a' switch to the next subsequent state is s'

CHAPTER 6 DESIGNING FEATURE-BASED OBJECT IDENTIFICATION ALGORITHM FOR THE BIPEDAL

Bipedal can do practically almost all necessary and basic errands assignments/ jobs, which are perilous and risky for a human being. To satisfy the above stated objective bipedal ought to have a visual framework that helps direct the bipedal about the routing. This framework/ system helps to recognize the objects and the controller of the bipedal would have the option to take desired actions. The algorithm is designed to handle vision-based navigation (VBN) of the bipedal. Bipedal distinguish objects by utilizing a revised SURF image detection algorithm. The bipedal controller has an action plan which helps in navigating in a risky and dynamic environment using the Q-learning RL algorithm. The bipedal segregates object depending upon already stored objects in the database and the objective for which the bipedal is designed to fulfill. The designed feature-based Q learning RL algorithm helps in decreasing the number of state values and helps in sharing and transmitting knowledge from one agent to another agent that uses RL to operate them. Likewise valuable for hurdle avoidance and recognizing hazardous articles during the exploring period.

6.1 Vision System in Bipedal Walking Robot

Bipedal can do intense and perilous assignments, which are dangerous for the human being. Bipedal helps humans in a risky environment - fire salvage activity, chemical ammunition. For performing such jobs basic issue is bipedal should have vision capabilities. This would help in identifying, detecting, and comparing objects by the bipedal.

Fast vision detection is carried out due to a decrease in cost and an increase in precision to manage data received by the navigational sensor. Vision systems are being supplanted by cameras with vision sensors for mission-critical applications. The bipedal first tallies the image captured online by image sensors with stored data then explores the dynamic environment. A set of unique features are derived using interest points from the present object image of the dynamic environment. At that point, these are coordinated with past knowledge on the highlights of the features of the stored object.

For comparing the captured image with a stored database, the algorithm ought to be invariant of scale and rotation of the captured image. Bipedal robots are autonomous, flexible, should be able to confront genuine circumstances, and should have the capability to see adjustments in the surrounding environment. The most crucial issue with bipedal is the selection of activities in the current scenario. When no specific model of the dynamic environment is available, Reinforcement Learning (RL) is used on bipedal. Through RL, the bipedal can figure out the outline of the map to be followed to move to the next state from the current state by selecting actions and computing the reward earned when associated with the dynamic environment. The main challenge of using RL in bipedal is enormous state-action space and vulnerabilities of a dynamic environment along with the online calculation of reward.

The control framework introduces information related to the state and stores online captured image data. The vision system of the bipedal captures images consistently from a dynamic environment (approximately every millisecond). The sequential frames are examined and the comparison of the current frame and the previous frame is carried out. If a distinction exists between the frames then distinct data is transferred to the image controller of the bipedal. The object recognition algorithm developed in the present research work is executed by the image controller. The recognition of the object is done. The further steps to be followed by the bipedal are dictated by the vision system in the form of information and the walking controller. The algorithm generally utilized is the Speeded Up Robust Features (SURF) algorithm.

6.2 Feature-Based Object Identification in Reinforcement Learning

Desire to know about the feature-based is due to the following reasons :

1) Need to decrease the quantity of state-space values to be managed by Q-learning Reinforcement Learning algorithm

2) We need to utilize these trained RL agents in an extensive dynamic environment in real life.

Advantages of utilizing Feature-based Reinforcement Learning algorithm - recognition of the object, planning the methodology/ approach as per need, handing over of information from one Reinforcement Learning agent to another Reinforcement Learning agent.

In the Q-learning RL algorithm, state-space values are dependent on some distinct features of a dynamic environment. Transferring of information from one RL agent to another RL agent about similar objects present and how to tackle them. For example, if in a dynamic environment there are bumps (speed breakers or rough terrain) and the height of bumps is fixed then what activities are to be taken to pass bumps to walk without falling can be shared between the RL agents. In the event of soccer, the robot needs to kick the ball recognition of soccer ball in the soccer field is the knowledge that can be shared between RL agents.

Two approaches to execute feature-based object identification are :

1) Simple encoding method, which is utilized to change the immediate environment of RL agent

2) Apply object recognition/ identification algorithms and then recognize objects.

Feature detection, extraction, and matching are steps that are usually carried out to solve machine vision problems. Computer vision problems are solved by object detection, object recognition, and content-based image retrieval (CBIR).

In the feature extraction step, a reduction in the dimensions of the image is effectively carried out. The result of this is a compact feature vector. This helps in rapidly matching and retrieving the images along with effectively reducing feature representation.

There are several considerations in choosing the number of features to extract:

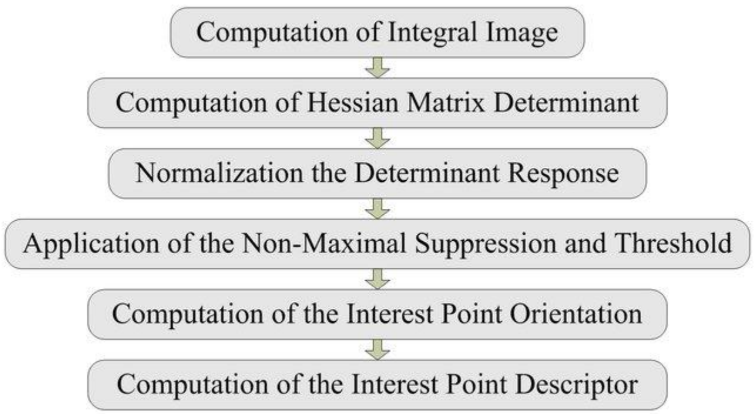
- More features use more memory and computational time.
- Fewer features can produce poor classifiers.

In this work, we have used a second approach to recognize objects. Object recognition is done by the revised SURF algorithm.

6.3 Comparative Study of Different Feature Extraction Algorithms

Image processing techniques include the operations on images like smoothing, sharpening, stretching, and contracting which results in an enhanced image that is usually used for image comparison in object classification and recognition steps. Object recognition has always been a computationally intensive job in real-time object recognition applications. The proposed method is an object feature detection SURF algorithm one of the image processing algorithms which is fast robust for local similarity invariant representation and comparison of images. This algorithm has three steps in a broader aspect to fulfill: interest point detection, local neighborhood description, and matching. These three steps include all the steps of other image processing algorithms: image preprocessing, image enhancement, image segmentation, image extraction, image classification. This algorithm uses a Hessian based detector, description based feature vector which is based on intensity distribution, using several approximations that allow fast computation without sacrificing accuracy and repeatability. The feature point of the stored image and the captured image are compared using a k-Nearest

Neighbor algorithm and a simple matching rule to identify the object in the environment.



Computation of Integral Image
Computation of Hessian Matrix Determinant
Normalization the Determinant Response
Application of the Non-Maximal Suppression and Threshold
Computation of the Interest Point Orientation
Computation of the Interest Point Descriptor

Figure 6.1 Detailed steps of SURF Image Processing Algorithm

The comparable algorithm for object feature detection algorithms is Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP). Before using SURF the comparative study of all the algorithms was done on clear and noisy both types of images.

SURF algorithm outperforms SIFT, HOG, LBP algorithms (Routray et al., 2017, Raj et al, 2017) on the complete data set. Performance of SURF algorithm doesn’t drop even in poor conditions like low light photographs, for photographs where only partial images of objects are found. SURF could extract up to 90%, the other algorithms could gain much less.

The results are compounded faster for SURF and LBP algorithms. The performance of SURF is close to SIFT and HOG.

Limitations with LBP, one of the oldest methods is the mean squared error increases and the performance gradually declines as the data sets get complicated and the algorithm cannot extract features completely for an object(Arunmozhi et. al, 2018).

SIFT is stable in terms of feature extraction but it gets slowed very gradually in feature extraction.

HOG shows its advantages in detecting edge and texture information of an image. The performance gradually decreases as the data set becomes complex. It can extract features but not as much efficiently as SURF and SIFT.

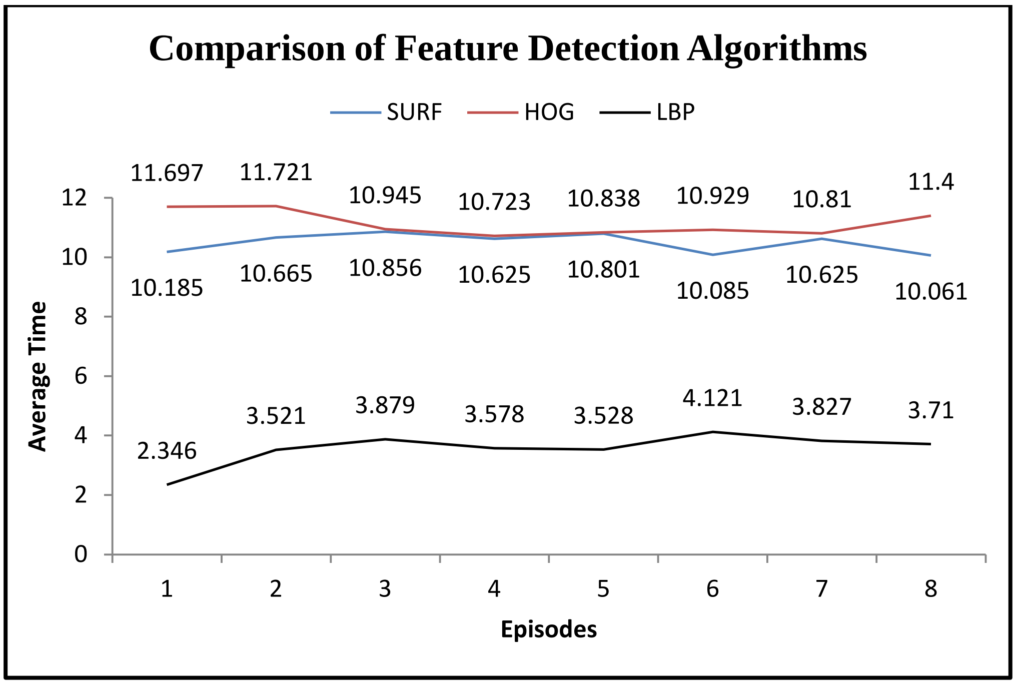


Figure 6.2 Comparison of Different Feature Detection Algorithm

6.3.1 Speeded-Up Robust Features (SURF) Algorithm

SURF is a powerful image detector/ locator and descriptor. Descriptor depends on the approximated Hessian model which gives the dissemination of Haar-wavelet reaction within the vicinity of interest points. Due to the low dimensionality of a descriptor, identifier and descriptor both diminish the hour of calculation. Speed, stability, uniqueness, and repeatability qualities of SURF make it a superior decision than other existing strategies. Interest point detection is done by Hessian matrix approximation. This determinant decides the scale and position of the descriptor. Box lets framework is utilized for fundamental images.

In steps to draw out SURF descriptor for an image - data based on the orientation of zone around interest points are utilized. These territories are round in nature, Haar wavelet is utilized to process directions in X and Y course summarizing Gaussian weights are utilized for horizontal and vertical reactions, maximum value characterizes direction of descriptors of interest points. Image scales are utilized as scale-spaces. Gaussian is utilized to smoothen images iteratively and sub examining results in reaching the next higher level of the pyramid. Continuously applying a filter is prevented by the utilization of basic images and box filters. Filter sizes are upscaled. Scale spaces are divided as octave, which is an arrangement of reaction maps. The filters are scaled in every one of the octaves by a scale factor of 2. The detection of interest points is done by indication of Laplacian, which helps in recognizing the bright spot on dark background and dark spot on a bright background. Quicker coordination is resulted in inspecting the points if they are on the same type of background (Lundberg et al., 2015; Okada et al., 2006). SURF has low dimensionality and reduces the time of computation as it executes faster.

Table 6.1 A Summary of State-of-Art Feature Detector

Category	**Classification**	**Methods and Algorithms**
Edge-based	Differentiation Based	Sobel, Canny
Corner-based	Gradient Based	Harris and its derivatives, KLT, Shi-Tomasi, LOCOCO,S-LOCOCO
Corner-based	Template Based	FAST, AGAST, BRIEF, SUSAN,FAST-ER
Corner-based	Contour Based	DoG-curve,ACI,Hyperbola Fitting etc.
Corner-based	LearningBased	NMX,BEL,SCG,DSC etc.
Blob(interest point)	PDE based	SIFT and its derivatives, SURF,LoG, CoG, RLOG,DART,KAZE,WADE etc.
Blob(key point)	Template Based	ORB, BRISK, FREAK
Blob(interest region)	Segmentation based	MSER. IBR,EBR,MFD,FLOG, BPLR

6.4 Proposed Algorithm for Feature-Based Object Detection

The proposed algorithm detects the image in the dynamic environment, then finds the correspondence between the captured image and the referenced image of the database. The image captured can be out of the plane, can be scale variant, or can have plane rotation. The algorithm takes care of all these before matching. The image captured is matched in the gray mode so that the matching process is fast(Sharma, Singh, Prateek, et al., 2019)

6.4.1 Stepwise Approach for Proposed Algorithm

Step 1: Read Images

- Capture the objects from the dynamic environment and extract the interested objects.
- Read the reference image from the database.

Step 2: Detection of Feature Points

- Identify the feature points (interest points) of both the images.
- Identify the strongest interest (feature) points from the reference image.
- Identify the strongest interest (feature) points from the captured image.

Step 3: Extraction of Feature Descriptors

- Extract feature vector using interest points in captured and reference images.

Step 4: Finding of Putative Point Matches

- Matching the features of both images using descriptors of each image.
- Display putatively matched features of both images.

Step 5: Localization of Object in the dynamic environment using Putative Matched

- Geometric transformation is done which helps in localizing the object in the environment.
- Eliminate outliers
- Display pairs of matching point after removal of outliers
- Display both objects
- A bounded polygon is obtained for the reference image.
- Polygon is transformed into the Cartesian coordinate system of the captured image. This transformed polygon helps in localizing objects in the dynamic environment.
- Display the detected object.

CHAPTER 7 DESIGNING OF REINFORCEMENT LEARNING CONTROLLER ALGORITHM FOR THE BIPEDAL

7.1 Reinforcement Learning: Introduction

RL is an AI technique (Stone & Sutton, 2001; Sutton & Barto, 2012) in which usually problems are solved which are goal-oriented in the dynamic environment. The process of making it stronger. RL is learning

1. What to do - situations are mapped with actions such that numerical reward signals are maximized.
2. The learner explores the actions which have maximum reward by trial and error method.
3. The selected actions affect the immediate reward as well as the next action to be taken along with all subsequent rewards.

A model of Reinforcement Learning consists of

- Set of state space of dynamic environment S which is a discrete set of state-space of environment S
- Set of action space A which RL agent can take (which is discrete)
- Set of reinforcement signals which is a scalar real value between {0, 1}.

The two major characteristics of RL are: searching by a trial-and-error method and delayed reward. RL is defined by characterizing a learning problem. In RL there is always a dilemma between exploitation and exploration. RL agent exploits previous state and action pairs which resulted in maximum rewards and explores new pairs to get better selection criteria for the near future.

Stepwise execution of basic model of RL

1. Agent receives as input ‘i’, which is at present state (s_1) in the dynamic environment
2. The agent selects an action from a set of actions (A) to generate output which is in step (3)
3. (a) The action taken by the agent transits to another environment state (s_2)

   (b) The state transition is informed to agents through a reward/ punishment signal (r)

Figure 7.1 shows the basic model of the reinforcement signal.

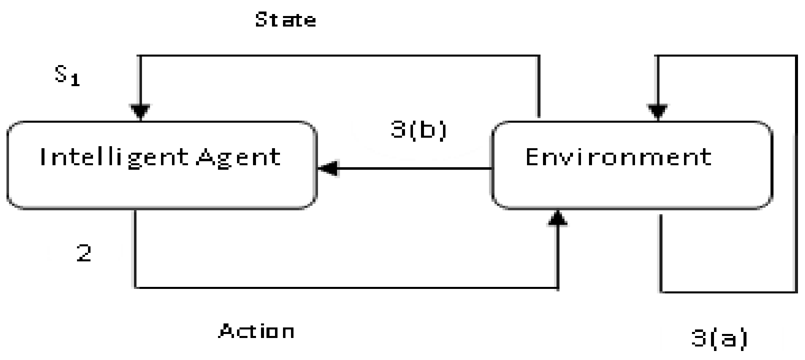


Figure 7.1 Basic Model of RL

Reinforcement learning is used when the environment is dynamic and uncertain and the agent finds the optimal policy. The agent continuously interacts with the dynamic and uncertain environment and gets feedback information which is processed through an appropriate algorithm to get the near-optimal/ optimal policy through which the agent explored. The environment in which RL learns can be model-free or model-based. In a model-free method, the controller is learned without a learning model whereas in a model-based method(Matarić, 1997; Ng, 2012) first the algorithm is learned then it is used to derive the controller. Some model-free methods are: temporal difference methods, Q-learning, average rewards, and model-based methods are: Certainty equivalent, Dyna, queue Dyna, priority sweeping(Sharma et al., 2013).

Four main subsystems of the RL system are:

a) **Policy**- reveals behavior at a specific time of the RL agent.

b) **Reward function**- An objective defined along with RL function is defined, which affects immediate reward. This is a short term incentive to the agent.

c) **Value function**- The reward selection depends on the action taken by the RL agent in the dynamic environment which affects the subsequent rewards of the agent. This is a long term reward forecast of the agent. These are also known as delayed rewards of the RL agent.

d) **Model of the environment (optionally)** - Resembles the dynamic and uncertain environment in which the RL agent operates.

RL is learning from a reward signal to choose an optimal (or near-optimal) action (a) in the present state **(s_t)** of the RL agent to optimize the reward (long-term) of the algorithm. There are algorithms for optimizing the finite horizon, un-discounted reward $V(t_0) = \sum_i^T r(ti)$, the (in)finite horizon discounted reward $V(t_0) = \sum_i^\infty \gamma_i(t_i)$ (γ is the discount factor) or also the average reward $V_A(t_0) = \lim_{T\to\infty} \frac{1}{T} \sum r(t_i)$, but the infinite horizon discounted reward is most commonly used. The agent learns from trial and error and attempts to adapt his action selection policy according to the received rewards.

The popularity of reinforcement learning is due to its unsupervised learning approach. The steps usually followed are - defining the reward function, then the RL algorithm is learned by choosing the optimal action policy which maximizes the immediate and the delayed (long-term) reward. It is not that easy as it sounds. There is a huge variety of RL algorithms.

Commonly used are value-based algorithms or policy search algorithms. The former agent learns from the expected discounted horizon reward for each of the next states while in the latter, the search is directly carried out in the parameters space of the action and next stable state. For policy search algorithms, any optimization algorithm can be used, so some approaches use genetic algorithms or simulated annealing to search for a good policy.

7.2 Reinforcement Learning: Reasons to use

1. Once the RL agent learns for a specific dynamic environment then it can use the previously gained knowledge, which helps in adapting to the system as time passes.
2. For a Model-free system, minimal help of experts is required for a model-based system more help of an expert is required who possesses application domain knowledge.
3. The agent learns in a very short span, and hence the solution is acquired.

7.3 Problems of Reinforcement Learning

In practice, a learning problem faces many restrictions to achieve an optimal/ near-optimal policy. Reinforcement Learning algorithms have the following issues:

1. **The curse of dimensionality**: Discretization of state space and action space is required. For some high dimensional control problems, discretization is practically impossible.
2. **Many trials for learning**: To train a system with huge state space and action space take a considerable amount of time, makes it a challenging task
3. **Finding algorithm parameters**: The parameters which directly affect the performance of the RL algorithm should be taken into account then those parameters are set so that algorithm runs with fewer parameters and give preferable results.
4. **Exploration - Exploitation Dilemma**: In the learning trials process, the agent will become stuck in suboptimal solutions, because the agent has not searched through the state space thoroughly enough. On the other hand, if too many exploration steps are used, the agent will not find a good policy at all.
5. **A skilled 'reinforcement learner' is needed**: Defining the reward function, and efficient state-space representation, or a good function approximation, choosing an appropriate algorithm, and setting reasonable

parameters of the algorithm are also important for effective execution of learning algorithm.

7.4 Multi-Agent System

Multi-Agent System (MAS) means a system in which many agents interact with each other in definite relationships and have different skills and knowledge about the dynamic and uncertain environment. Each agent has a sensor, motor, knowledge base, and learning component. Multi-Agent System uses Artificial Intelligence, intelligent control, computer technology, and sensor technology (optional).

These components of agents incorporate some restricted activities :

1. The sensor component can know a limited environment.
2. The motor component is specialized in performing only a specific set of actions.
3. There can be an incompatibility between actions carried out by different agents of the MAS.
4. Multi-Agent System (MAS) or Finite State machine for this work is a Bipedal Walking Robot.

7.5 Various Reinforcement Learning Algorithms

7.5.1 Temporal Difference (TD) Learning Algorithm

The Learning Agent learns through every single action it takes rather than on every episode or reaching the goal or end state(Sharma et al., 2013).

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate] \quad (7.1)$$

The value of (Target - OldEstimate) is called Target Error. StepSize is α called learning rate whose value lies between 0 and 1.

Temporal Difference (TD) learning is a way to learn how to predict a value depending on the future values of a given state.

Q-learning is a specific way of TD learning for learning Q-values.

7.5.2 Q – Learning Algorithm

Q-learning is an off-policy model-free algorithm Reinforcement learning algorithm. The model-free means the agent learns through experience rather than the old available experienced data. This helps in handling stochastic elements and a large sequence of state-action pairs. The learning agent does not have any idea about the transition system and the rewards awarded. The agent has to interact with the dynamic and uncertain world.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\ [R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \qquad (7.2)$$

Q-learning takes the optimal path. It assumes that the agent is following the best possible policy without attempting to resolve what is the actual policy. In Q-learning greedy policy is used. The main goal of Q-learning is to maximizing the Q-value or use a method that optimizes Q-value.

7.5.3 SARSA (State-Action-Reward-State-Action) Algorithm

SARSA is an on policy temporal difference control method. A policy is a state action pair tuple that helps in mapping action to be taken at each state. An on policy control method is applied by letting the agent transition from one state-action pair to another state-action pair. SARSA take a safe path means it explores less and exploits more.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\ [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \qquad (7.3)$$

SARSA looks ahead to the next action to see the next step and update the Q-values of its current state-action pair accordingly. SARSA takes the agent's actual policy into account.

7.5.4 Deep Q – Network (DQN) Algorithm

A deep Q network leverages a neural network to estimate the Q-value function. The input for the network is the current while the output is the corresponding Q-value for each of the actions. Q-learning has the drawback that it does not have any clue which action to take if the agent has not visited that state before this problem can be overcome by DQN.

$$r_j + \gamma max_{a'}\, Q\left(\varphi_{j+1}, a'; \theta^-\right) \qquad (7.4)$$

The loss function for the network is defined as the squared error between the target Q-value and the Q-value output from the network

7.5.5 Deep Deterministic Policy Gradient (DDPG) Algorithm

Deep deterministic policy gradient relies on actor-critic architecture: actor and critic. An actor tunes the parameter θ for the policy function i.e. decides the best action for a specific state.

$$\pi_\theta(s,a) = P[a|s,\theta] \tag{7.5}$$

A critic is used for evaluating the policy function estimated by the actor according to temporal difference (TD) error.

$$r_{t+1} + \gamma V^v(s_{t+1}) - V^v(s_t) \tag{7.6}$$

Lower case v denotes the policy that the actor has decided**.**

7.6 Why Q –Learning Method?

Reinforcement Learning and Queue Learning (Q-Learning) are readily being used in robotics for navigation and exploration of the dynamic environment. The most widely used Q-learning algorithm is simple, efficient, and is very adaptive to the uncertain environment. Due to this Q-learning is ideal for robotics navigation. The aim of current work includes bipedal should learn goal-oriented navigation strategy and to learn the shortest path to reach goal state considering obstacles and current coordinates of an object in the dynamic environment. The search algorithms are modified to suit the way to solve the problems in a dynamic environment. The main goal is to maximize Q-learning or optimize the policy or choose the greedy policy(Sharma et al., 2013).

The bipedal learn the reward function from the task model from repeated trails. A modified RL algorithm with a forgetting mechanism that optimizes speed and memory consumption is proposed and implemented in the MATLAB platform.

Table 7.2 Comparison of Reinforcement Leaning Algorithms {(Mahadevan,1996);(Tesauro ,1995)}

	TD Learning	**Q-Learning**	**SARSA**	**DQN**	**DDPG**
Learning Method used	Learns at each action taken	Handle Problems with stochastic transition and rewards	Learns with action performed by the current policy	Learns by minimising the loss	Learns by Q-learning and Policy gradient method
Policy Type	Off	Off	On	Off	Off
Model Based/ Model Free	Model Free RL	Model Free RL	Model Based RL	Model Free RL	Model Free RL
Agent dependency	Independent of agent	Independent of agent	Dependent on agent	Independent of agent	Independent of agent
Learning Policy	Learns Optimal Policy with the help of greedy policy	Identify optimal policy for maximising total rewards over all successive steps	Learns from current action, current state, reward, next action, next state.	Trains function approximator and uses ε-greedy policy	Learns by actor critic model
Calculation Methods	Estimates rewards for future actions and new state is appended without actually following any greedy policy	Computes with the maximum expected rewards for an action taken in a given state	Takes current state and action to estimate	Optimal policy is feed to current state into optimal Q-function, takes action which maximizes all future actions	Actor tunes θ paramneter and critic uses to estimate the policy function by TD error

7.6.1 Q – Learning Algorithm

The original Queue Learning is a simple incremental algorithm that was designed keeping in mind the dynamic programming for delayed rewards. In the Q-learning algorithm, a two-dimensional lookup table is used which is indexed by state-action pairs. The bipedal is designed using Markov Decision Process (**MDP**).

An **MDP** is an ordered group of <State(S), Action(A), State Transition probability(T), Reward Function(R)>.

State S - The set of states should be finite including the start state and terminal state.

Action A -Finite set of actions, available actions depends on the current state of bipedal.

State Transition function T- Convey probability $p(s'|s, a)$ that bipedal will move from current state *s* to next optimal state s' when *an* action is taken from

Reward function R - gives immediate reward $r(s, a, s')$ real value given to bipedal when bipedal takes action *a* while transiting from state *s* (current state) to s' (next state). Reward signal is in the form of encouragement/ punishment to bipedal.

S_t states at time t, a_t is the action taken by S_t at *t* time, *r* is an immediate reward received by bipedal when action *a* is taken and system transits from present state S_t to the next stable state S_{t+1}.

Q-learning (Rezende et al., 2014; Sandon et al., 1956) iteratively approximate value function Q which tracks state-action. The learning of policy and the value function is carried out simultaneously. The RL algorithm is designed for the three joints of the leg of the bipedal. MATLAB platform is used to develop this algorithm. The algorithm is learned offline and online. Q-learning algorithm (Watkins, 1989)(Watkins & Dayan, 1992)

```
Initialize Q (s, a) Arbitrarily;
Repeat (for each episode);
    Initialise s;
    Repeat (for each step of episode);
        Choose a from s using policy derived from Q;
        Take action a, observe r, s';
        $Q (s, a) \leftarrow Q (s, a) + \alpha [ \gamma \max a' Q (s', a') - Q (s, a)]$;
        $s \leftarrow s'$;
    until s is terminal,
until all episode's end.
```

Figure 7.2 The Q-learning Algorithm

7.7 Reinforcement Learning (RL) Model: Stepwise Approach

Kinematics and dynamics analysis gives real-time information to the reinforcement controller of a bipedal robot. If any singular position comes during the movements of the joints, it is bypassed by the trajectory control of the joint. In the reinforcement control algorithm(Feil-Seifer & Matarić, 2008; B. Q. Huang et al., 2005; Y. Huang et al., 2013; Martinez-Cantin et al., 2009; Sternberg & Kaufman, 2016), the action value selection depends on maximum and minimum values of acceleration of joints. The magnitude of time required to take action can be too small or too large to switch to the next state. RL does not state the method in which the task is to be carried out. In RL, agents are programmed by rewards and punishments. The bipedal senses the next state with the help of a gyroscope sensor, which gives real-time orientation feedback about the current position of the joint to the RL controller. The reinforcement controller's selection of the next action is dependent on the current state. Further corresponding leg joint is moved to the next stable state. This process is iteratively carried out until the goal/ target point is reached by the bipedal. Figure 7.3 shows the basic RL model of the bipedal walking robot. The present work considers the dynamic environment, bipedal fall in the forward direction and reverse direction. RL algorithms are designed and developed for the hip, knee, and ankle of the bipedal robot of each leg.

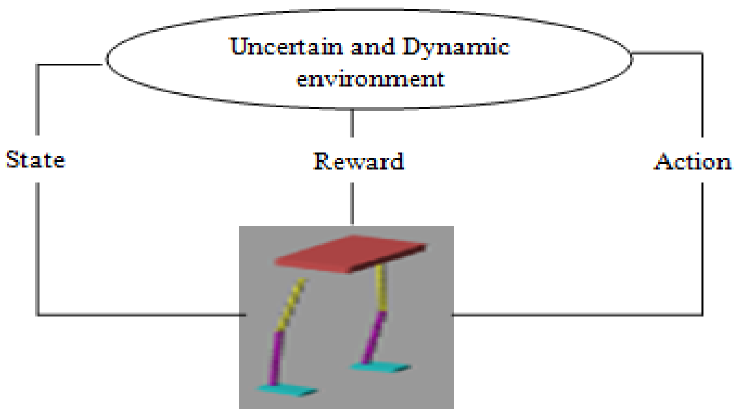


Figure 7.3 Basic RL Model of the Bipedal Walking Robot

After sensing the current position, the signal will process by the reinforcement controller. The controller looks at the look-up table and compares the next state of the system. During switching the system sometimes fails to reach the desired state. In reinforcement learning, the learning parameter decides the accuracy of the system(Sharma et al., 2020)(Bharadwaj et al., 2019).

7.8 Forgetting Mechanism incorporation into Traditional Q- learning

RL agent when interacts with a dynamic environment sometimes attempts to utilize prior learned knowledge. This expertise may be outdated as the environment is dynamic and uncertain which results in a change in the dynamics of the environment. This may change the exploration and exploitation dilemma.

To restrict the use of previously learned knowledge which may be outdated due to change in dynamics of environment incorporation of forgetting mechanism is done in traditional Q-learning algorithm.

In a deterministic environment, Q-learning is simple as there is exist mapping for subsequent states and actions. The state values associated with each action rather than with each state-action pair are stored. In a dynamic environment, state-action pairs and the dynamic reward which is calculated in real-time are stored(Sharma, Singh, Bharadwaj, et al., 2019).

The reward function is maintained by the State value function. The state-value function is initialized to zeros. When the RL agent explores an uncertain environment, it learns rewards associated with each state of a dynamic environment. After some steps, it evaluates a tradeoff between the exploitations of new states, action pair, and exploitation of choosing actions that have previously resulted in near-optimal/ optimal policy. The Q-value function is a two-dimensional lookup table that is updated after each run a particular state is visited. (Figure 7.2) Q-learning is a model-free RL method.

7.9 Action Selection Policy

The generation of a random number between (0,1) results in the selection of action policy. In the first go, if the value lies in the range (0,0.5), exploitation of action selection takes place. If the value lies in the range (0.5,1), the exploration of new action from the set of actions is done. For subsequent goes the exploitation range is modified to (0,ε*ε-decay) and the exploration range is modified to (ε*ε-decay,1). RL favors exploration rather than exploitation which reveals the dynamic nature of the environment. RL agent is trained not to get stuck in the same state for a long time. ε controls policy updation which is based on the next state selection. For the current work, ε is assumed to be 0.5, it is usually a scalar value (0,1). Updation is done as ε=ε* ε-decay after each episode.

If ε→1 shows more dependency on the following state and so less forgetting occurs. The algorithm behaves like a traditional Q- learning algorithm.
If ε→0 shows large dependency on functions of state transition and reward calculation and almost all previous rewards are forgotten between episodes. The exploration of the dynamic environment by the RL agent is carried out relatively in each episode, hence previously learned knowledge is not utilized.

CHAPTER 8 SIMULATION OF REINFORCEMENT LEARNING ALGORITHMS FOR BIPEDAL

The stepwise approach to achieve the desired objectives:

1. Observe the current state of the bipedal robot. This information is sensed by sensors which give exact orientations of all three joints along with the stable standing position of the bipedal.(Ghavamzadeh & Mahadevan, 2007; Lee & Labadie, 2007; Mahadevan, 1996; Palmer, 2007; Schwartz, 1993; Watkins & Dayan, 1992)
2. Identification of object using feature-based extraction by the bipedal robot in its path of movement. The object in this case is a soccer ball on the path. The bipedal has to identify the object based on its feature and using an updated SURF algorithm along with the affine transformation. After identifying an object in the path bipedal finds its exact location or Cartesian coordinates in the dynamic environment.
3. Action selection policy, joints are actuated by DC servomotors which move in full or no speed in both directions. Pulse modulations help in controlling the speed of the motor. Bipedal is not a preprogrammed robot, so deciding an action is dependent on the discounted factor (λ), epsilon (ε), learning rate (α), epsilon-decay, and some random values.
4. Performing action. RL algorithm runs for discrete state values, the time taken to reach the goal state from the current state depends on the capabilities of the processor. Practically, no mathematical equation exists for time calculation. Hence, the time required by the bipedal to reach the goal point depends on probabilities and randomness to reach to next stable step. This depends on the action taken and the current state of bipedal.
5. RL reaches the next stable state when an optimal/ near-optimal action is performed.

6. Calculation of random rewards is done when after selecting some optimal action the bipedal moves to the next stable state. When bipedal successfully achieve the next stable state, the positive reward is assigned for that action, and the next state is selected. The reward is dynamic in the current work and is calculated on the fly. The random reward is calculated by finding the distance from the present state to the goal state then multiplying the result so obtained by learning rate (α is considered 0.9 in this work) then taking the negative exponential of the evaluated value and then assigning it to the immediate reward. If the goal is not reached, then a negative reward can be assigned. In current work, no negative rewards are calculated to state and action as creates the problem in the proposed work. But when the bipedal robot is falling on the ground, to stand up on both feet. Robot to stand upright position needs to move both the feet either in the forward direction or reverse directions.
7. Bipedal learns from experience. The reinforcement controller calculates the distance from the current state to the goal state by finding the difference between them. If the distance comes out to be zero then the bipedal reached the goal state and stops. If the distance is not zero then bipedal repeats sequence from step two until the goal point is reached. The optimal actions and policies are stored in the lookup table so that it can be used in the future if the same scenario exists.
8. The bipedal moves near to the soccer ball and should kick it in any direction (if possible). Hence the objective is accomplished.

8.1 Stepwise Approach to Desired Objectives (as Incorporated in Algorithms)

Step 1: 1. Observing the current state of the bipedal robot by sensing with the assistance of the sensor.
2. It is giving the actual/ real directions about the position of the ankle, knee, and hip joint.

Step 2: 1. The bipedal robot has to identify a feature-based object in its path of movement.
2. The object, in this case, is the soccer ball on the path

3. Identify the object based on its feature using the SURF algorithm along with the affine transformation.

4. After identifying objects i.e. soccer ball, find their exact location.

Step 3: 1. Involves deciding action.

2. DC servomotor is utilized to activate the joint.

3. A not preprogrammed robot, so deciding an action depends on the rate of learning(α), discounted factor(λ), epsilon-decay, epsilon(ε), and some randomness values.

Step 4: 1. Involves performing an action.

2. RL algorithm is substantial for the discrete state between present state to goal/ target state.

3. Time is taken to arrive at this state relies upon processor capabilities.

4. No scientific conditions are required to compute the time.

5. Users cannot control the time to reach the bipedal robot to reach the goal point.

6. It depends absolutely upon the randomness and probabilities to reach the following step.

Step 5: RL observes the new state by performing the action.

Step 6: 1. RL calculates random rewards.

2. Bipedal robot when successfully reaches the following state, a positive immediate reward assign to action and state. (Negative reward are not considered)

3. When the bipedal robot is falling, stand up on both feet. It must move the feet either in expedite/ invert direction to recover standup position.

Step 7: 1. Involves learning of bipedal robot from the experience of the current run of the RL algorithm.

2. Stores optimal actions, optimal policy in the lookup table which helps if the scenario of the dynamic environment is the same in the future.

3. If the goal point is accomplished by the bipedal robot, reinforcement controller separation between current state and goal state if zero then stop at that point

3. If it is not achieved the goal then repeat the sequence from step three until the goal point is reached

Step 8: 1. Involves the movement of the bipedal near to the soccer ball

2. Kick it in any direction (if possible)

8.2 Model of Proposed Framework/ System

The proposed framework has two principal parts where processing is done. They are Feature processing and Q-learning RL algorithm. Feature processing incorporates feature extraction, feature matching, and object identification. Resultant processed state(s_p) is then contributed to the Q-learning algorithm then the action is captured in the dynamic environment resultant is following/next state (s_{t+1}) alongside dynamic reward generation(r_{t+1}). The following state(s_{t+1}) is then contributed to feature processing through deferment and the reward generated (r_{t+1}) is input to the Q-learning RL algorithm through a deferment(Sharma, Singh, Prateek, et al., 2019).

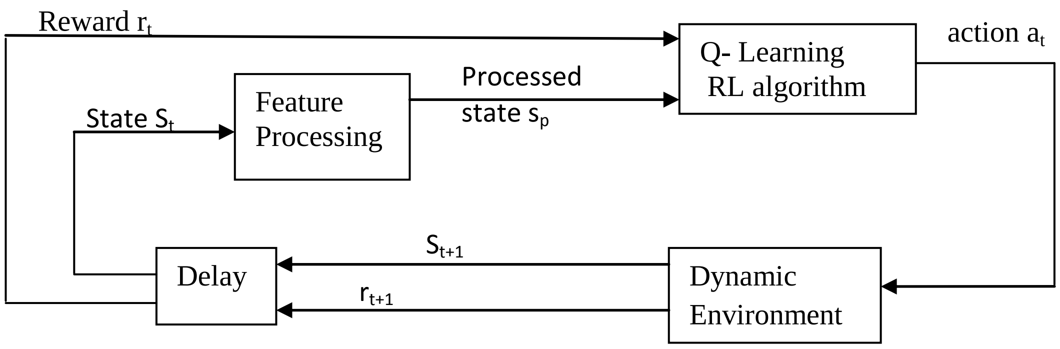


Figure 8.1 Model of the Proposed System (RL agent)

8.3 Implementation of Q-Learning Algorithm

In the current scenario, the research society is attempting to design a self-selecting proficient robot for a dynamic environment. In pre-programmed robots, the controller is carrying out the responsibility in a known environment. But, with the presence of a dynamic environment, because of the absence of decision-making capabilities, these sorts of controllers neglect to carry out the responsibility. In such cases, pre-programmed bipedal doesn't have the foggiest idea of what to do. The reinforcement algorithm assists in carrying out the task/ responsibility. The reinforcement algorithm(Stone & Sutton, 2001; Sutton & Barto, 2012) manages the current situation with the bipedal robot. It is detecting the current state of the bipedal robot and taking the bipedal to the next state without knowing the mechanics and dynamics of the framework. So it is a model-free based controller.

The reinforcement control algorithm is completely autonomous from the mechanics and kinematics of the body of the bipedal robot. But while picking action it is incompletely reliant on dynamic qualities. While choosing action to move to the next stable state, torque on joints is changing. Now the joint motor is fit to deal with these torques. There is some pecularity point while taking steps towards the following state.

Pecularity/ Singularity is where the controller doesn't have the foggiest idea of what to do. (Details description in Appendix B and C) So there is a need to sidestep that point, otherwise, the controller is not sending any data to the controller to do the next job. The reinforcement control algorithm is dependent on value function, transition probabilities, and cost function reward. The reward function is a blend of positive and negative qualities. In the present work, the positive value of the reward is awarded when the controller is arriving at the subsequent step effectively. A negative reward isn't awarded to action since when bipedal falls on the ground, to reach to the goal position, bipedal ought to bring both feet either in a forward way or reverse way. If a negative reward is granted to state and action, next time feet aren't doing an action to carry feet in a reverse direction. The current work is centered around the balance/ dependability of the lower body of a bipedal robot. When some slipping condition occurs due to a change in environment, at that point of time robot isn't capable to come in an underlying state i.e standing position of a bipedal robot.

The reinforcement control algorithm assists with acquiring robots in standing conditions without any preprogramming of robots. The stability of the walking pattern of the bipedal is situated on the convex hull formed between the ground and the feet.

8.4 Proposed Algorithm for Incorporating Forgetting Mechanism

The proposed algorithm proceeds as (Sharma, Singh, Bharadwaj, et al., 2019) -

1. The environmental parameters are decided: epsilon(ε), learning rate(α), epsilon decay, discount factor(λ)
2. Q matrix is initialized to zero
3. For every run -
   A. Selection of Initial state is conducted (initially received by sensors, at runtime evaluated by an algorithm)

   B. Do while target state has not been reached

      a. Select one among the potential actions for present state utilizing random value generator (Exploit/ Explore)

      b. The random reward is computed, using this conceivable action

      c. Moving to the following state is examined, using this conceivable action

      d. Maximum Q value for the following state is evaluated, which is dependent on all possible actions

      e. Process Q(s,a) value

      f. Write intermediate results, current state, action selected, following state, the total time for execution, immediate reward in an excel file

      g. In lookup/query table store the optimal actions, policies, and the next state so that can be utilized in future

      h. For every run, intermediate results are stored in a new excel sheet

   end do

   C. Express final Q-matrix, optimal strategy, total reward computed, mean random value generated, total time taken in another excel file.

   D. For each episode, final results are stored in a new excel sheet.

4. Store values of each episode and the number of iterations required to reach the target state along with the total duration for each process in the third excel file.
5. Plot graphs for comparison of the number of iterations in the learning and execution phase along with total time for execution in each episode, total reward awarded in each episode, mean random value generated in each episode.

8.5 The Proposed System's Characteristics

1. Storing and managing the number of states, action taken and next state separately decreases memory space. Resulting in a more effective learning algorithm.
2. When the agent explores it learns reward/ penalty related to every state which is approximated by the value function of that state.
3. After every step
   - A. Selection of actions depends upon the generation of random values
   - B. Computing reward/penalty
   - C. The next state is assessed with the assistance of the current state and determines the reward
   - D. Revision in Q-value (Policy) for visited state
   - E. Epsilon recalculated by epsilon*epsilon decay
   - F. Computing total rewards/penalty
   - G. Computing total execution time
   - H. The separation between the current state and the goal state assessed
   - I. How far is an agent from the goal state is tracked

As the agent explores, the value of states arbitrarily remote from goal will move towards maximum value i.e. reward(max)/penalty(max), Q(max), and for states close to the goal will move towards a minimum value i.e. reward(min)/penalty(min), Q(min).

Assuming that after the vast period of exploration, all states arbitrarily remote from goal will accomplish the same value V(max) = $\frac{rewards(max)}{1-\lambda}$ and V(min) = reward(min).

8.6 Reinforcement Controller

8.6.1 Model Free Controller: Reason to Use

Model-free reinforcement learning controllers do not rely on any specific mathematical model of the system. These controllers do not rely on the stored experience but the online values calculated by the bipedal from the dynamics of the uncertain environment. The RL controller is solely based on online measurements collected directly from the dynamics of the bipedal and the environment. The bipedal finds an optimal policy that acts as a strategy used by the bipedal to behave in a dynamic environment(Sharma et al., 2020; Sharma, Singh, Prateek, et al., 2019).

In a stochastic environment, if the bipedal takes an action in a certain state, the resulting next state of the environment might not necessarily always be the same. These uncertainties will make the task of finding the optimal policy harder. The bipedal has to deal with the uncertainties in the environment and decide the next action to be taken on the fly as the rewards are calculated and the distance between the current state of bipedal and the goal state (position of the soccer ball) is also calculated on the fly. This justifies that model-free controllers estimate the optimal policy without using the transition and reward functions of the dynamic environment. The Q-learning update rule also does not have any term of probabilities only the rewards which are being calculated on the fly and are not fixed to any constant value.

8.6.2 Reinforcement Learning (RL) Controller

A reinforcement controller is executed in the MATLAB platform as shown in Figure 8.2. Details of the Simulink block is described in Appendix F
The bipedal robot decides on its own by knowing the current state and switches over to the following state without knowing the kinematics and

dynamics of the framework. In the current research work, the lower body of the bipedal robot is at an assured location and that location is 0° of ankle joint, -25° of the knee joint, and -15° of the hip joint for the left foot and the opposite of these values to the right foot.

When these values are detected with the assistance of a gyroscope sensor, a bipedal robot needs to venture to every part of the goal point. The goal point for the ankle joint, knee joint, and hip joint are 20°,20°, and 15° individually. The time taking to arrive at these points is independent of the kinematics and dynamics computation. In reinforcement controller time taken to reach the next state, is reliant on processor abilities and generating the control signal. The reinforcement controller controls the intermediate position of the joint between the initial point and the goal point.

Figure 8.3 shows the interaction of the reinforcement controller to the lower body of bipedal.

The randomness of the controller helps to take the action. The Q-learning algorithm allows the bipedal to go from one state to another state. The reward is gain during the switching by the state-action pair. After several running, the state-action pair of the reinforcement controller keeps the updated values. In the next time of running the system executes with the old values and tries to switch to the next state.

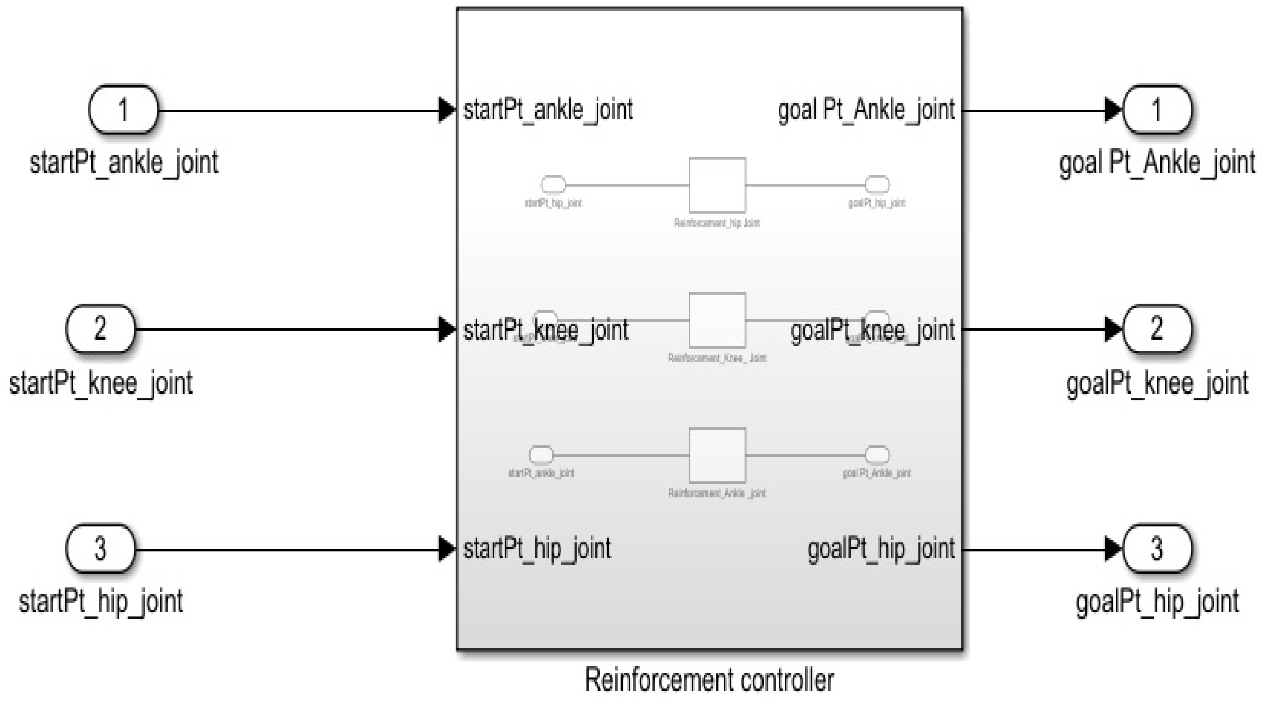
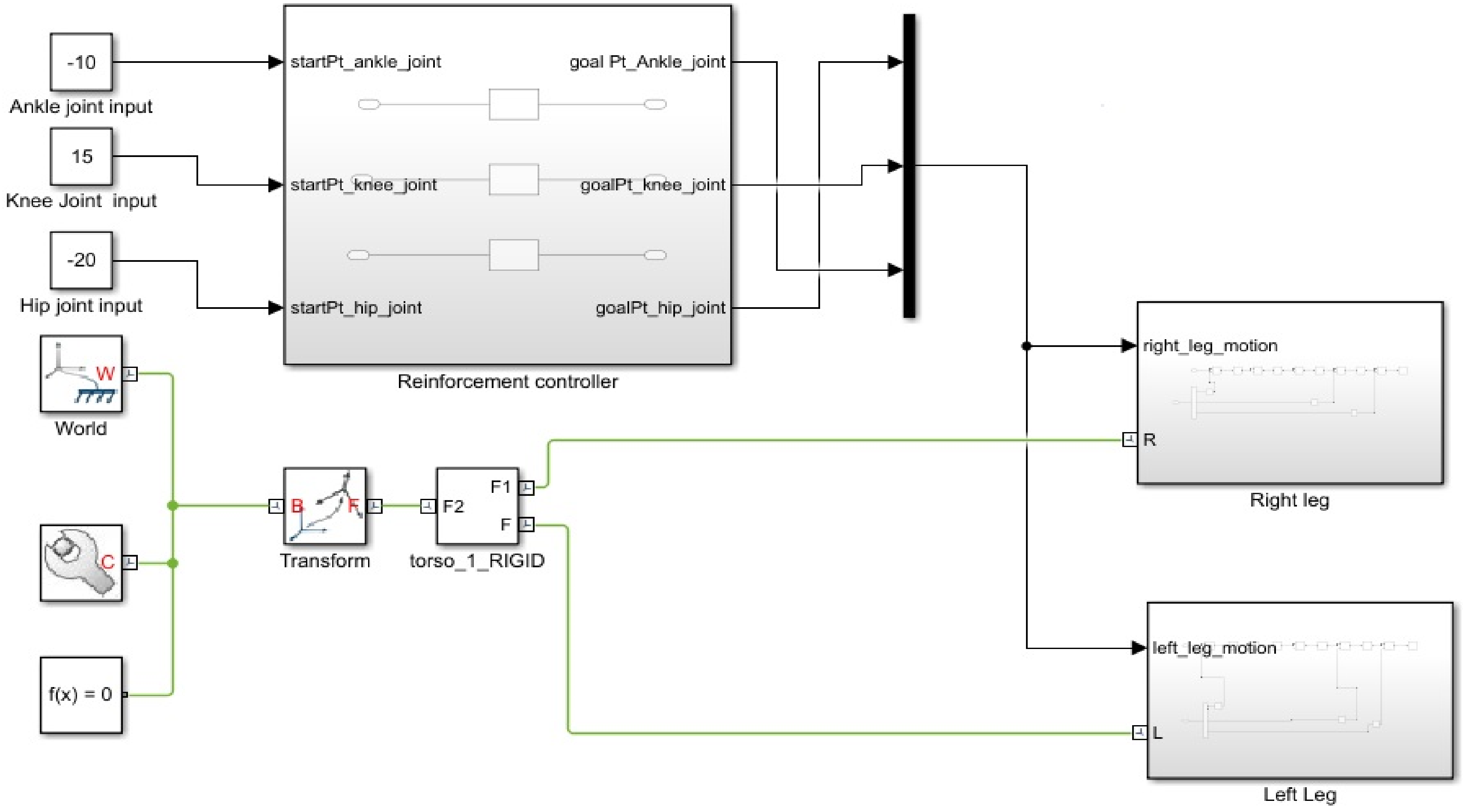


Figure 8.2 Simulink Block Diagram of Reinforcement Controller



-

Figure 8.3 Interfacing of Reinforcement Controller to Lower Body of Bipedal

CHAPTER 9 PROGRAMMING, TESTING, AND VALIDATION OF DESIGNED ALGORITHM

Simulation is carried out for the proposed model as shown in Figure 9.1 in the SimSpace Multibody dynamics toolbox. Figure 9.1 shows an initial state of the lower body of the bipedal. After sensing the current state, the lower body is switching to a goal state without knowing any kinematics and dynamics of the present system.

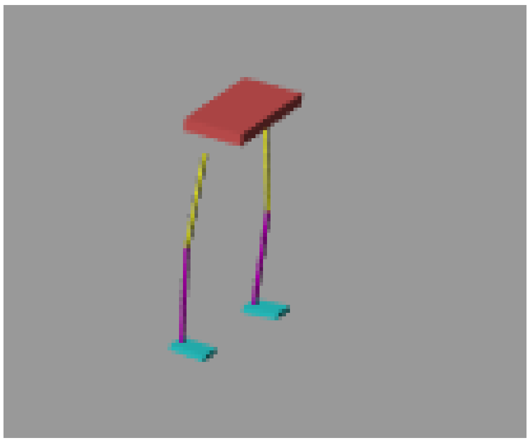


Figure 9.1 Bipedal Robot is at Current State

9.1 Experimental Findings of Forgetting Q-learning Algorithm

The simulation of the bipedal was carried out and while learning and execution results are stored for future study. Firstly, the bipedal was learning in which exploration was the main aim while in execution exploitation was the aim.

Two distinct ways of storing the data are:

1. Graphical representation of results as graphs and
2. Store data in the lookup tables for future use.

The algorithm is characterized by

1. Q(s,a) updation rule
2. The function which evaluates action to be taken
3. Forgetting Mechanism
4. Randomness in reward/ penalty depending on the current state of RL agent and goal state(Sharma et al., 2020)ras.

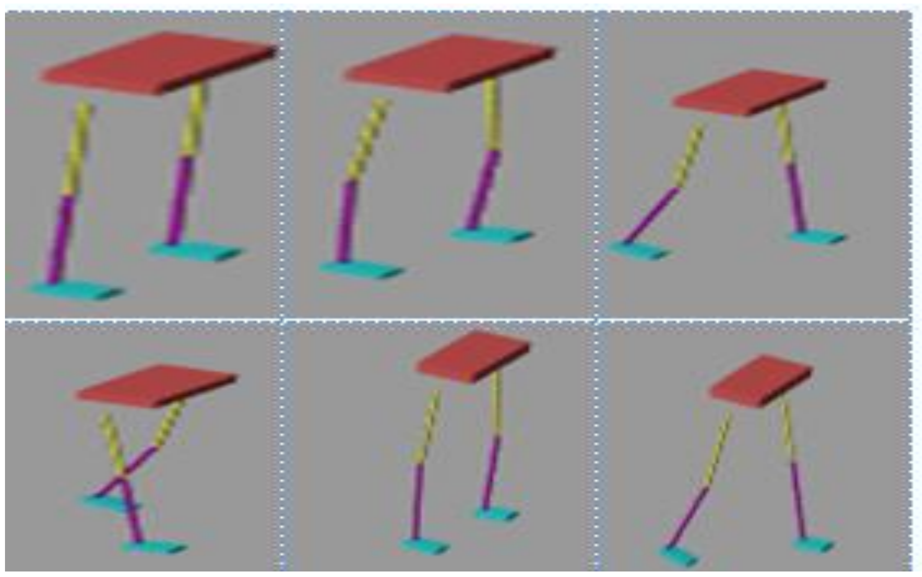


Figure 9.2 Locomotion of Bipedal Robot (Model 1)

The objective is to provide upgraded execution in the dynamic condition by using exploratory conduct that keeps a larger set of possible solution then is kept by the traditional Q-learning algorithm.

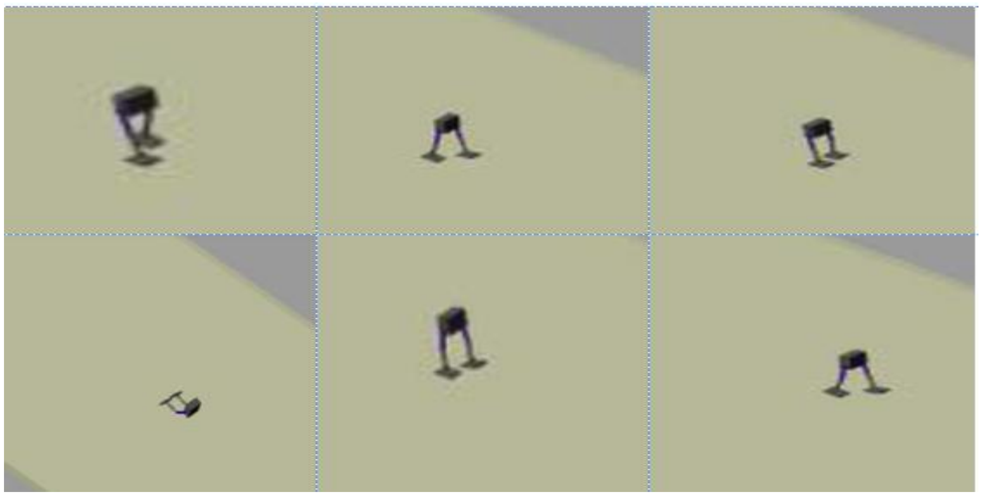


Figure 9.3 Locomotion of Bipedal Robot (Model 2)

In model 2, (Figure 9.3) hip, knee, and ankle joint were less prompt while the locomotion so slight changes in the diameter of the upper and the lower leg were done for the bipedal (Figure 9.2)

9.1.1 Simulation Results

Case I- Random value generated between (0,1) Singh, Bharadwaj, et al., 2019)

The intermediate, final, and graph data of the algorithm is saved in three different lookup tables along with a header for future use.

The intermediate lookup table contains state, action transitions along with penalty/ reward, a random value generated in each episode along with the time required to run each episode, and distance between the current state of RL agent and goal state which RL agent has to reach.

The final data lookup table has a mean value of random values generated to reach the goal state from starting/ initial state, final Q(s, a), the optimal policy obtained along with actions selected in optimal policy, total reward/ penalty to reach initial state to goal state along with total time.

The graph data lookup table contains the number of iteration, mean random value, mean reward generation, the total time required to execute each episode. (Figure 9.4 and Figure 9.5)

The graphs are plotted for

1. Target state - current starting state of RL agent in each episode
2. Target state - next state in each episode
3. Mean random value and random value of each episode
4. Distance between the current state of RL agent and goal state
5. Reward/ penalty in each episode

ε value ($=\varepsilon* \varepsilon$-decay) in each episode incorporates the forgetting mechanism.

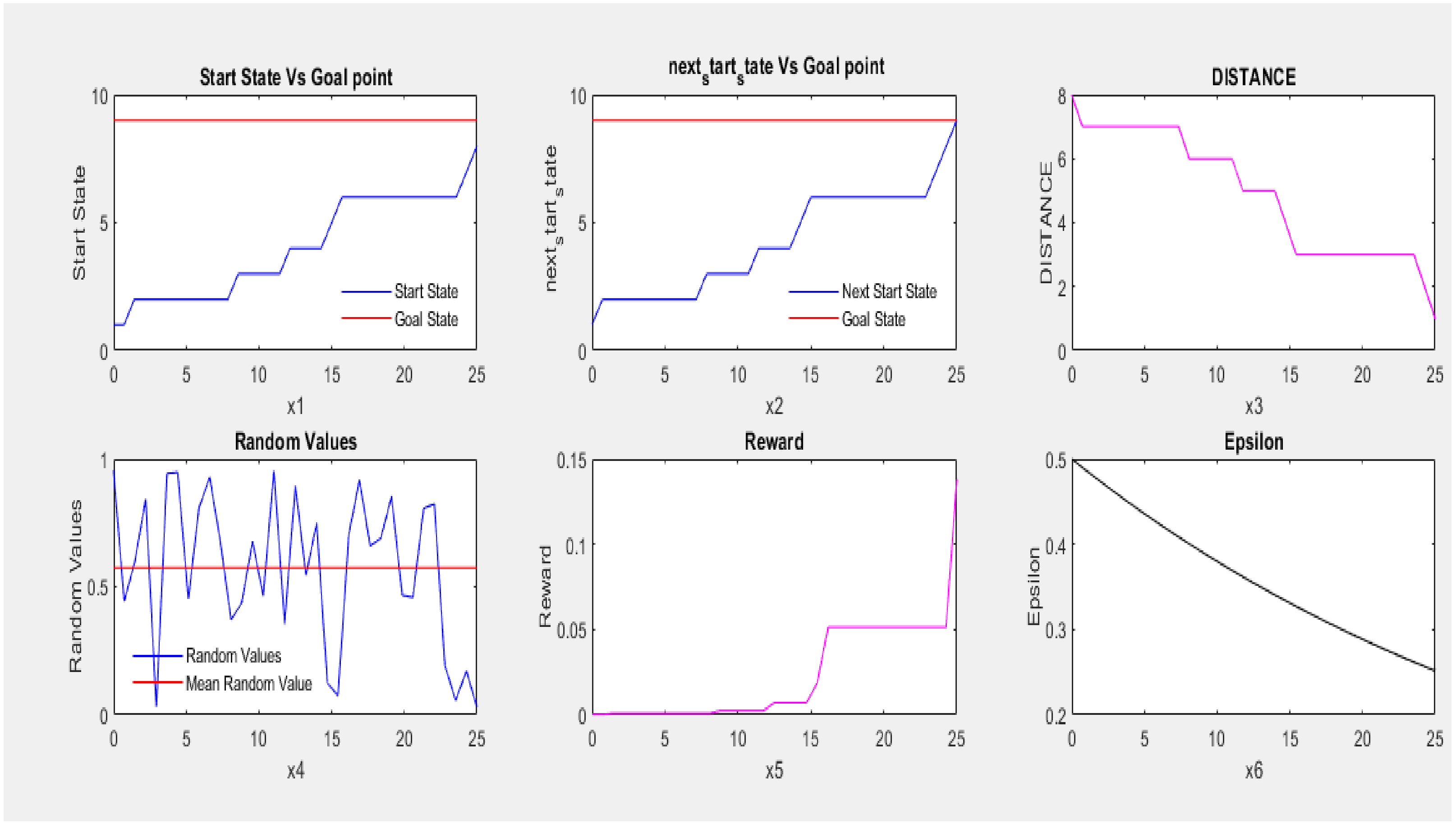


Figure 9.4 Case - I Reached the Goal State in 36 Iterations

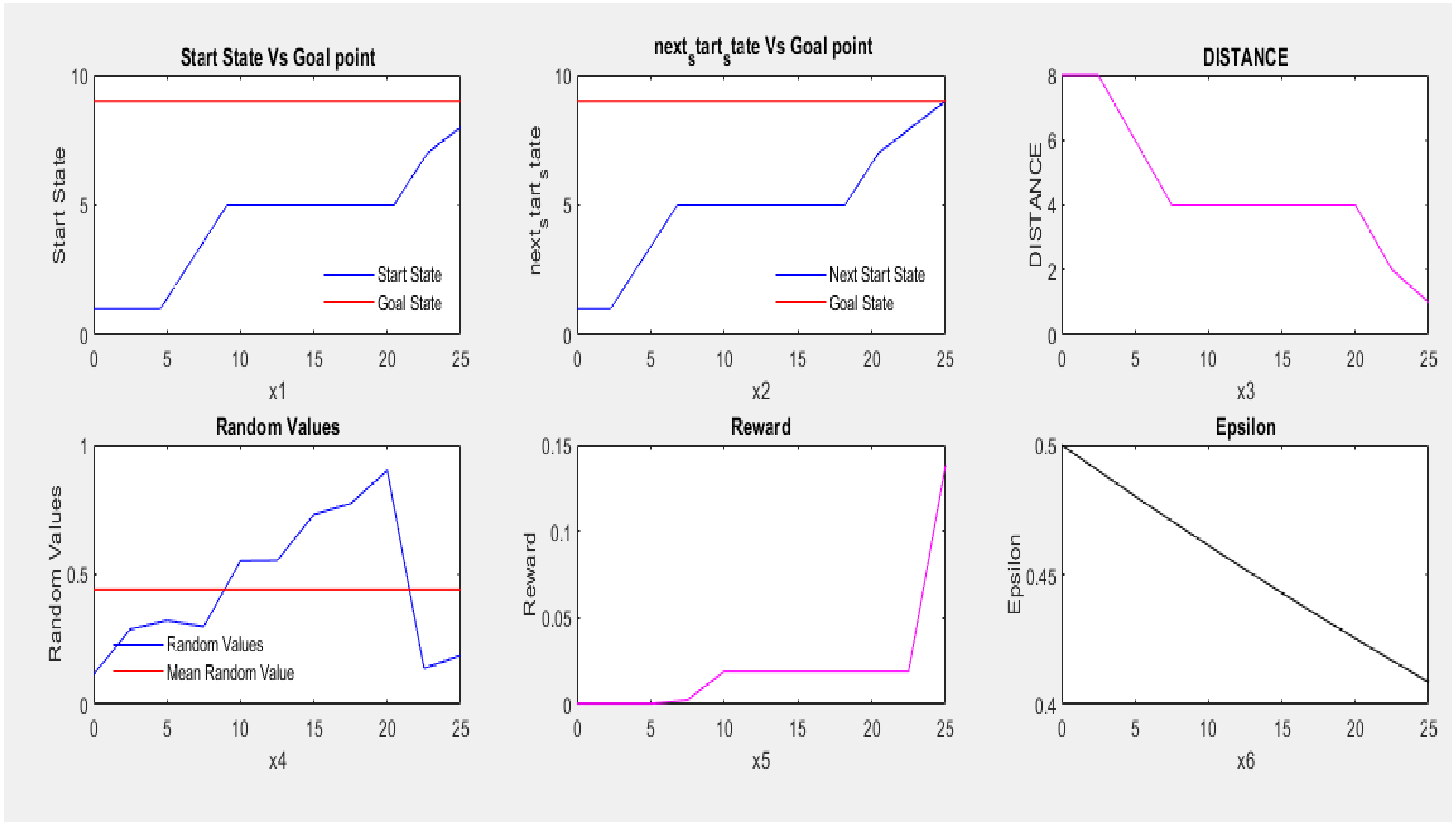


Figure 9.5 Case I- Reached the Goal State in 12 Iterations

CASE II When the range of random number generation is fixed (0.4 to 0.6)
With the same state set and action set (known set) and all parameters the same as the previous simulation, it is observed that it runs the maximum number of episodes but does not reach the goal state but gets stuck in between the start state and goal state. This is due to the reason that action of 0 degree is considered due to which bipedal remains in a specific state for an infinite duration. (Figure 9.6)

CASE III - When considering action deg(0) does not reach the goal as it is stuck in the same state loop.

Without considering action deg(0) algorithm converges fast and it explores more as compared to exploit and does not fully exploit decay factor (Forgetting mechanism) (Figure 9.7)

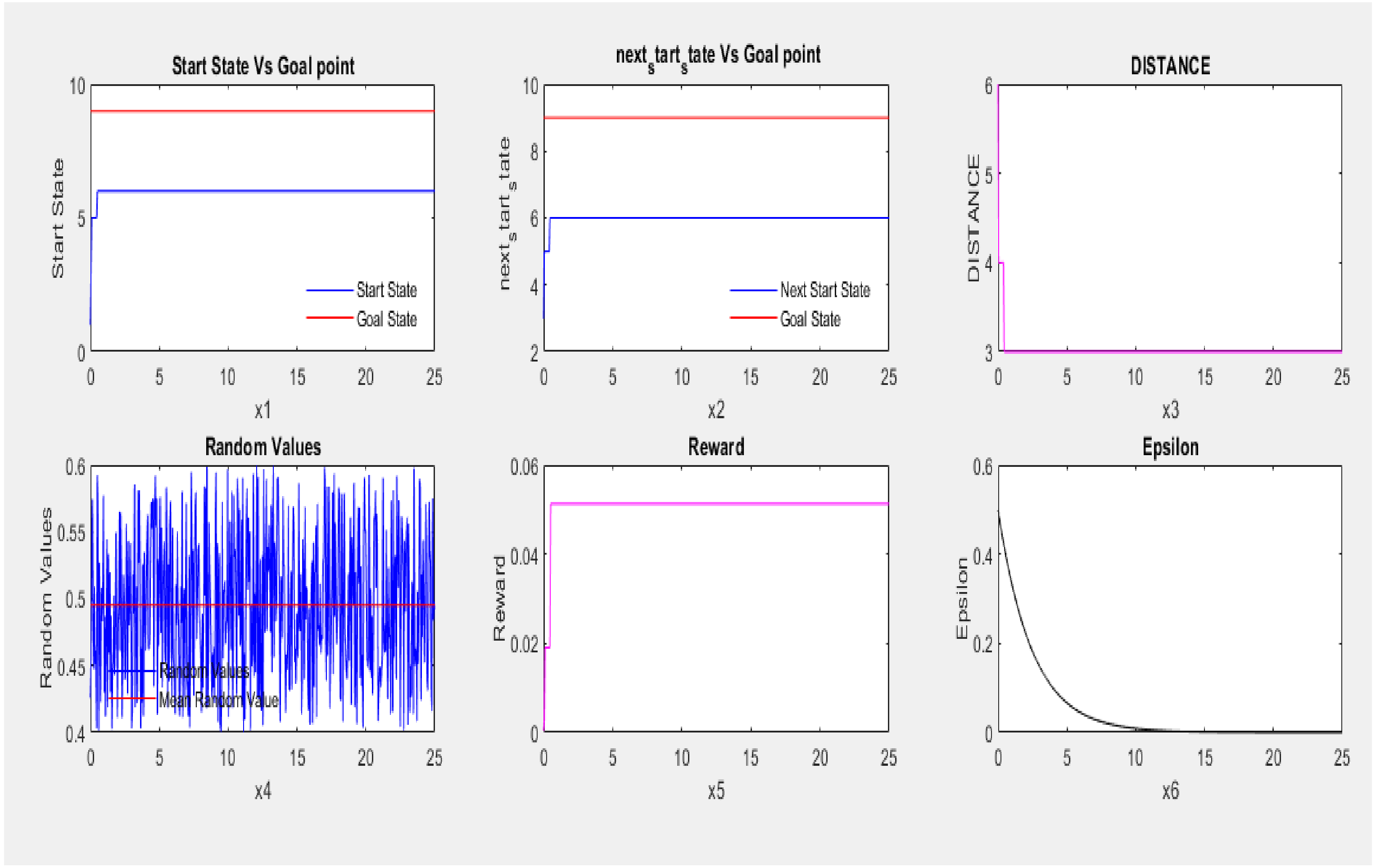


Figure 9.6 Case II - Does not reach Goal State even after Executing 100 episodes

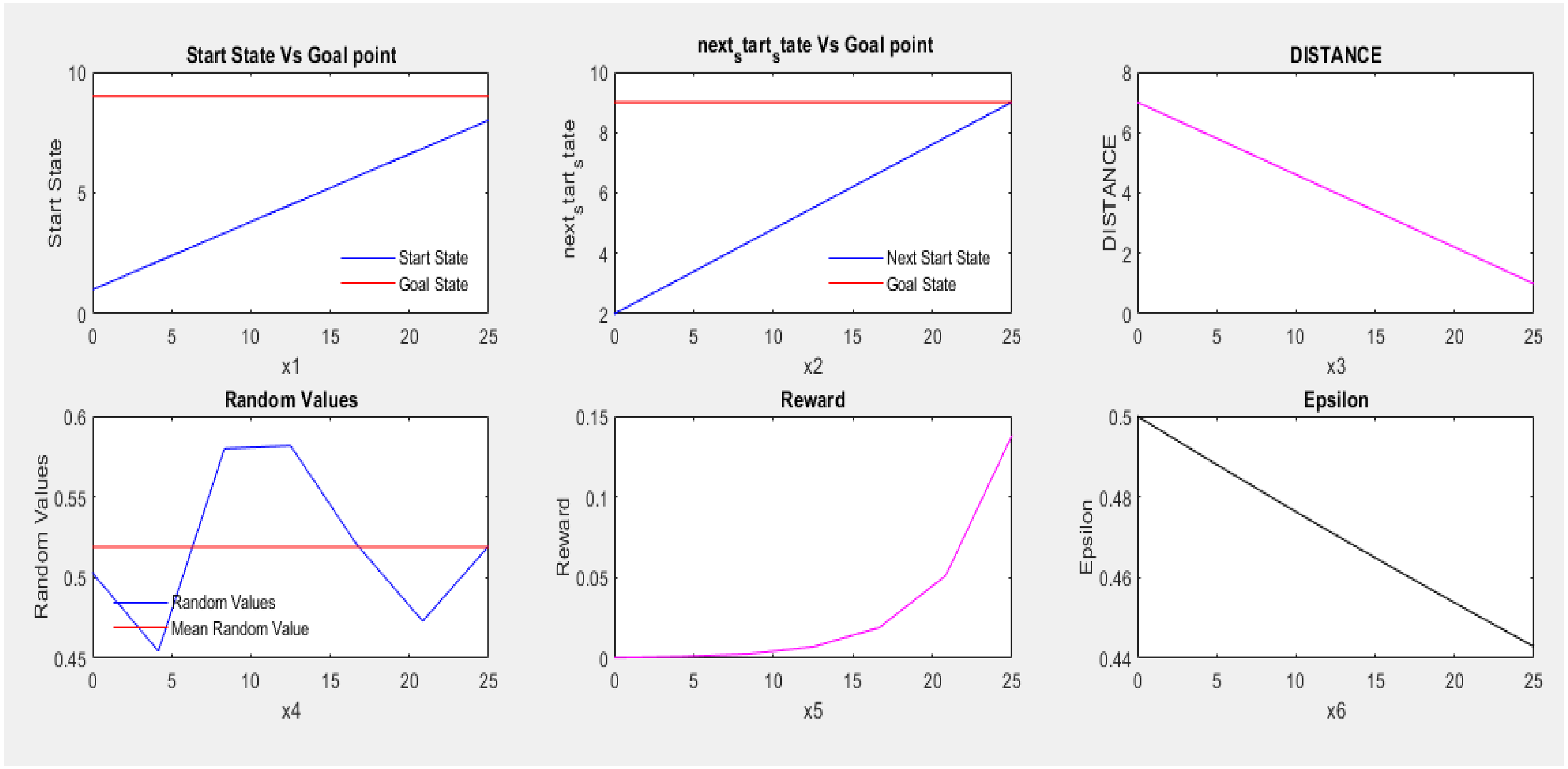


Figure 9.7 Case III - Reaches the Goal State in 8 Iterations

9.1.2 Data Saved in Lookup Table for the future use

Results found after simulations are stored in the lookup tables. Each episode stores its value in a different sheet which contains - current state of each episode, next state, random values generated, series of actions taken to reach goal state, list of immediate rewards along with delayed rewards, and total time required to execute current episode and the final result of each episode are stored for future use. (Table 9.1 - Table 9.6) (Sharma, Singh, Bharadwaj, et al., 2019).

Table 9.1 Case I Final Lookup Table

Mean Random	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Total Reward	Total Time
0.574685	0.00036	0.00036	0.00036	0	0	0	0	1	0	1	0	0	0.8248	0.2299
	0.00595	0.00097	0.00595											
	0.00898	0.00261	0.00898											
	0.01902	0.00701	0.01902											
	0	0.01887	0.01887											
	0.35006	0.05079	0.35006											
	0	0.13669	0.13669											
	0	0	0											
	0	0	0											

Table 9.2 Case II Final Lookup Table

Mean Random	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Total Reward	Total Time
0.44249	0.00068	0	0.00068	0	0	2	0	0	0	0	0	0	0.2562	0.15004
	0	0	0											
	0	0	0.00261											
	0	0	0											
	0.07744	0	0.07744											
	0	0	0											
	0.13669	0	0.13669											
	0	0	0											
	0	0	0											

Table 9.3 Case I Intermediate Lookup Table

Random Number	Current state	Current Action	Next State	Reward	Epsilon	Time	Distance
0.954086903	1	1	1	0.000363	0.49	0.030433	8
0.444338165	2	2	2	0.000363	0.4802	0.083759	7
0.599817138	1	2	2	0.000978	0.470596	0.098299	7
0.842621831	1	2	2	0.000978	0.461184	0.100825	7
0.031200453	1	2	2	0.000978	0.45196	0.105586	7
0.943594399	1	2	2	0.000978	0.442921	0.107787	7
0.947922139	1	2	2	0.000978	0.434063	0.11089	7
0.452984391	1	2	2	0.000978	0.425382	0.11394	7
0.810832551	1	2	2	0.000978	0.416874	0.117054	7
0.928879201	1	2	2	0.000978	0.408536	0.120004	7
0.672717281	1	2	2	0.000978	0.400366	0.122202	7
0.37233228	2	3	3	0.000978	0.392358	0.125802	6
0.438823922	1	3	3	0.002632	0.384511	0.129029	6
0.678649201	1	3	3	0.002632	0.376821	0.130739	6
0.465070329	1	3	3	0.002632	0.369285	0.132765	6
0.953264168	1	3	3	0.002632	0.361899	0.135408	6
0.354697982	2	4	4	0.002632	0.354661	0.138923	5
0.895860047	1	4	4	0.007083	0.347568	0.141278	5
0.545434258	1	4	4	0.007083	0.340616	0.146281	5
0.749283717	1	4	4	0.007083	0.333804	0.149378	5
0.124877199	2	5	5	0.007083	0.327128	0.166131	4
0.074749438	2	6	6	0.019063	0.320585	0.171523	3
0.703651642	1	6	6	0.051303	0.314174	0.176905	3
0.918950568	1	6	6	0.051303	0.30789	0.181696	3
0.660071908	1	6	6	0.051303	0.301732	0.187919	3
0.690104919	1	6	6	0.051303	0.295698	0.194076	3
0.85372361	1	6	6	0.051303	0.289784	0.200103	3
0.467901723	1	6	6	0.051303	0.283988	0.202879	3
0.458478682	1	6	6	0.051303	0.278308	0.206439	3
0.806104187	1	6	6	0.051303	0.272742	0.209979	3
0.824767236	1	6	6	0.051303	0.267287	0.213536	3
0.190436103	1	6	6	0.051303	0.261942	0.21704	3
0.05681496	1	6	6	0.051303	0.256703	0.220601	3
0.171418738	2	7	7	0.051303	0.251569	0.224403	2
0.029515208	2	8	8	0.138069	0.246537	0.229863	1

Table 9.4 Case II Intermediate Lookup Table

Random Number	Current state	Current Action	Next State	Reward	Epsilon	Time	Distance
0.114789	1	1	1	0.000363	0.49	0.043864	8
0.289081	1	1	1	0.000363	0.4802	0.052354	8
0.323706	3	3	3	0.000363	0.470596	0.062168	6
0.299634	3	5	5	0.002632	0.461184	0.070919	4
0.552771	1	5	5	0.019063	0.45196	0.100299	4
0.554692	1	5	5	0.019063	0.442921	0.103989	4
0.730647	1	5	5	0.019063	0.434063	0.108803	4
0.773625	1	5	5	0.019063	0.425382	0.113474	4
0.900847	1	5	5	0.019063	0.416874	0.124663	4
0.138167	3	7	7	0.019063	0.408536	0.144453	2
0.189413	1	8	8	0.138069	0.400366	0.150035	1

Table 9.5 Case III Intermediate Result Sheet

Random Number	Current state	Current Action	Next State	Reward	Epsilon	Time	Distance
0.503135	1	2	2	0.000363	0.49	0.010173	7
0.454399	1	3	3	0.000978	0.4802	0.060253	6
0.579907	1	4	4	0.002632	0.470596	0.068851	5
0.581739	1	5	5	0.007083	0.461184	0.074326	4
0.520729	1	6	6	0.019063	0.45196	0.104072	3
0.473047	1	7	7	0.051303	0.442921	0.111539	2
0.519718	1	8	8	0.138069	0.434063	0.11716	1

9.2 Experimental Results of Feature-Based RL Agent

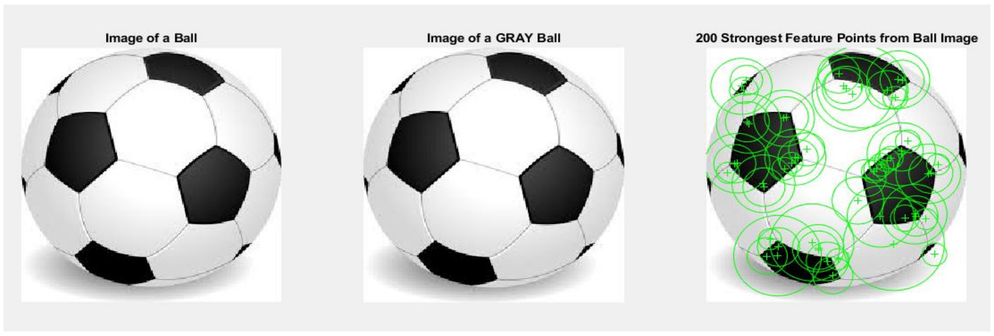


Figure 9.8 Soccer ball with 200 strongest points identified

Table 9.6 Case III Final Lookup Table

ean Random	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Total Reward	Total Time
0.518954	0.00036	0	0.00036	1	1	1	1	1	1	1	1	1	0.21949	0.1172
	0.000968	0	0.000968											
	0.002606	0	0.002606											
	0.007013	0	0.007013											
	0.018872	0	0.018872											
	0.050790	0	0.050790											
	0.136689	0	0.136689											
	0	0	0											

Figure 9.8 shows an image to be compared to a soccer ball. The colored image is converted to the gray image than 200 strongest points were identified which help in feature matching(Sharma, Singh, Prateek, et al., 2019).

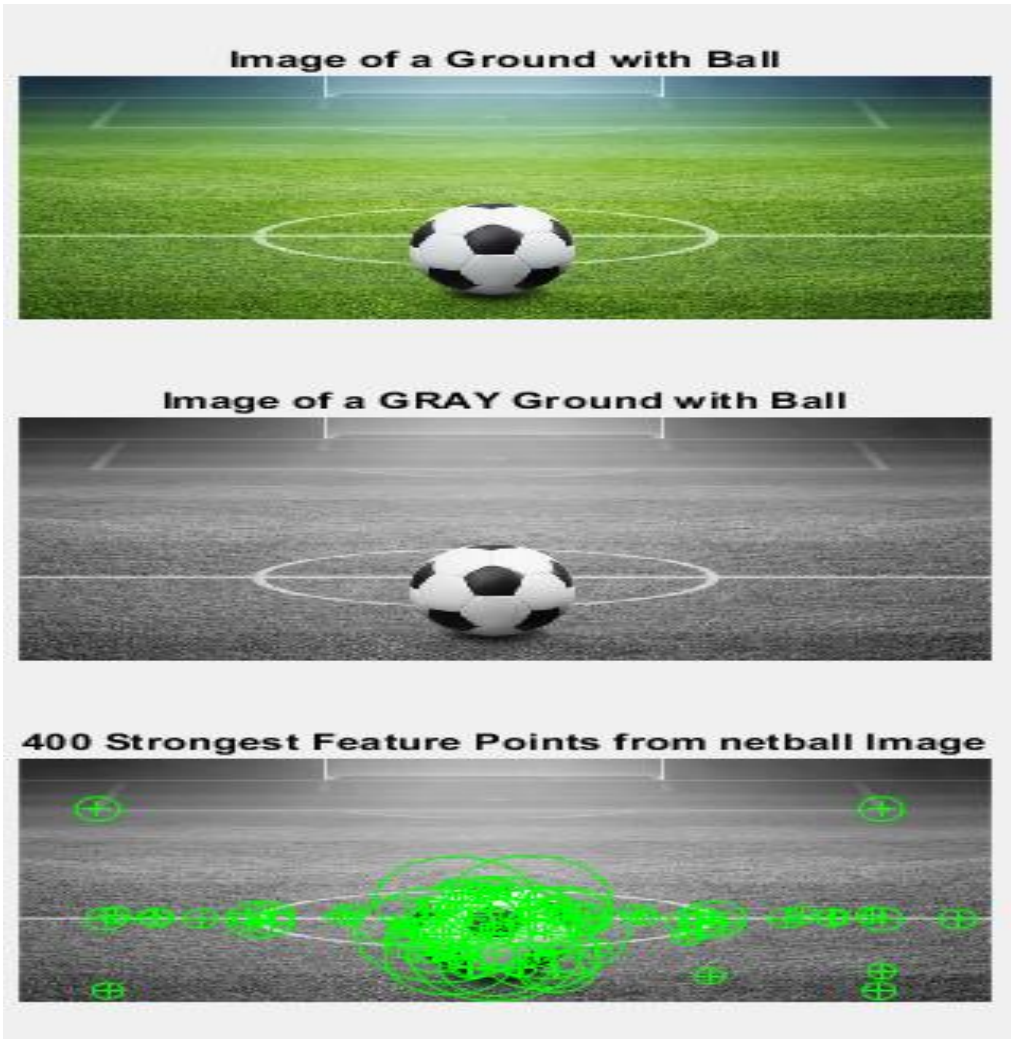
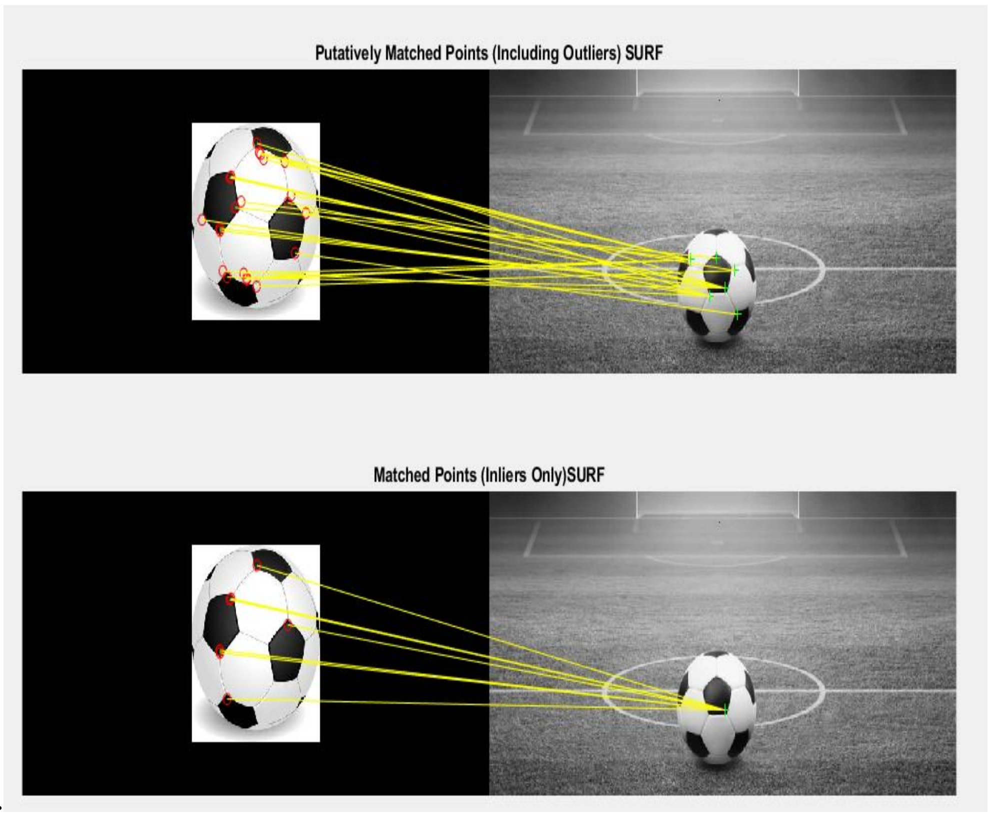


Image of a Ground with Ball
Image of a GRAY Ground with Ball
400 Strongest Feature Points from netball Image

Figure 9.9 Soccer ball on the ground, its gray image and 400 strongest points identified

Figure 9.9 shows the image of the ground where the ball is present. The colored image is first converted to the grayscale image than 400 strongest points were identified which help in feature matching. The strongest point identification reveals that the ball has more points identified and less on the ground.


Putatively Matched Points (Including Outliers) SURF
Matched Points (Inliers Only)SURF

.

Figure 9.10 Top - Matched Points including Outliners
Bottom - Matched Point including only Inliers

Figure 9.10 matches the feature obtained from Figure 9.8 and Figure 9.9. The left side is Figure 9.8 strongest point image and the right-hand side is Figure 9.9 strongest point identified image. These two images are affine feature matched top figure shows including outliers on SURF algorithm and lower figure shows including only inliers on SURF algorithm.

Figure 9.11 shows ball identification on the ground in both gray and colored images. Hence, the feature is processed, and the result states (position) of a soccer ball on the ground. The soccer ball is identified on the ground. The position of the ball is calculated in Cartesian coordinates and passed to the Q-learning RL algorithm. The controller then sends an instruction to bipedal joints to move the bipedal robot to reach the soccer ball accordingly(Sharma, Singh, Prateek, et al., 2019).

9.3 Hierarchical Structuring of RL System

Training the hip joint for the forward movement is done then the knee joint is trained using forgetting mechanism Q-learning algorithm then the ankle joint is trained similarly considering the contact forces of the feet too.

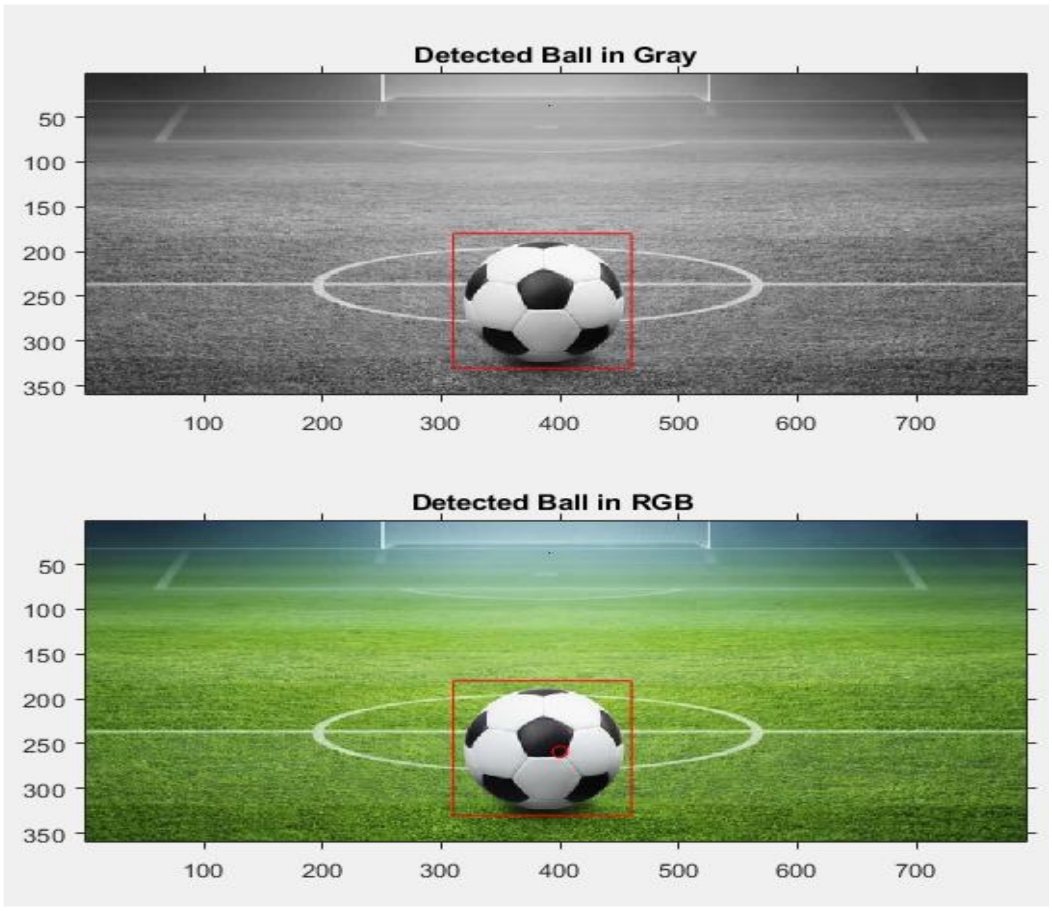


Figure 9.11 Soccer Ball identification done in Gray and Color Image

The bipedal is trained, in first-run Bipedal walks with jerks as seen in each of the joints and takes more execution time i.e. more iterations of the proposed training algorithm. But when a simulation is executed repeatedly then the trajectory of the bipedal is smooth and moves fast. As the Gait cycle is fixed it reaches a goal point at the approximately same time. Previously when the simulation started the learning of the bipedal is being carried out.

For a few steps of the learning phase, the optimal data is stored in the lookup table. The optimal action, next state, rewards, and optimal policy values are stored in the excel file. After the completion of the learning phase, these data are further utilized for further execution of the bipedal in the same dynamic environment. This results in a considerable amount of reduction in the number of iterations. In the learning phase, the maximum number of iterations was 51-52 or 46 depending upon the joint and the initial and goal state and eventually reduced to 21-23 as a minimum after the learning phase was completed. In the execution phase, the maximum number of iterations is 18 and eventually reduces to 2 as a minimum, This is so by using the optimal action and the next state values in the lookup table rewards are calculated on the fly.

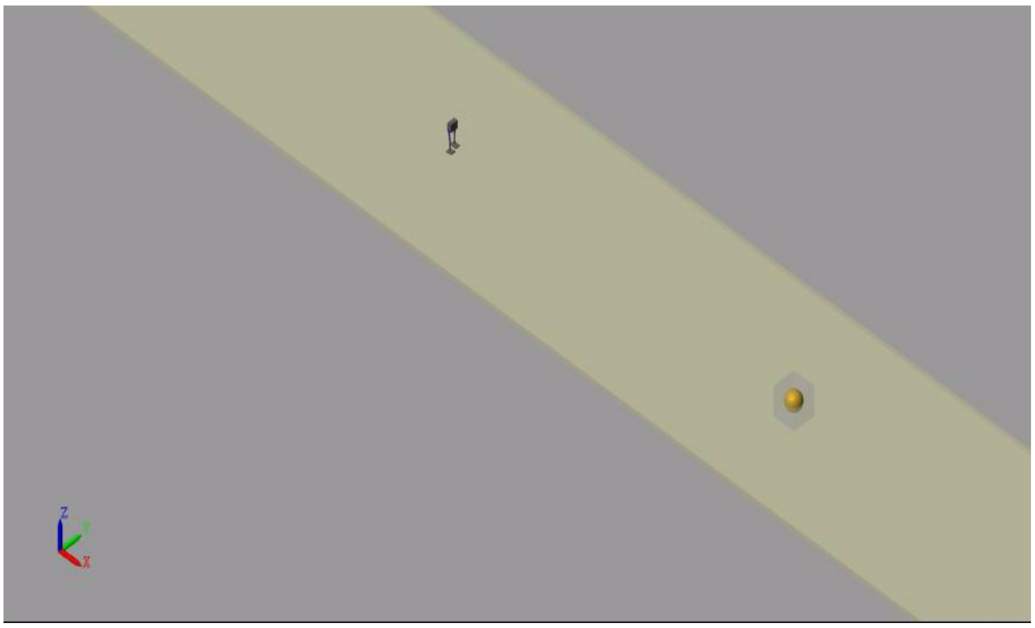


Figure 9.12 Locomotion of Bipedal Robot after Object Identification

9.3.1 Proposed Assumption

At the beginning of the research, the learning phase was based on the number of strides or the number of episodes to be executed but later it was based on the range of the number of iterations, which when reaches a minimum range (21-25) the learning phase or the exploitation stage of bipedal comes to end. This leads to the starting of the execution phase or the exploitation stage where the bipedal uses the stored data in the lookup table if the scenario of the dynamic environment is the same. The same scenario is justified by the object identifications in the dynamic environment, the Cartesian coordinate of the object is calculated when these are the same as one stored in the database, then only the execution phase has to be carried out using the stored lookup data.

When Cartesian coordinates stored vary then the scenario is a new one and has to start from scratch means to continue the learning phase(Sharma et al., 2020).

In the learning and execution phase, the main concerned area is the maximum and a minimum number of iterations in both these phases along with the total time required in the learning and execution phase of each episode/ stride.

9.3.2 Learning Phase

In the learning phase, learning is done starting from 25 strides, 50 strides, 75 strides, 100 strides, 150 strides, 200 strides. The trends observed in these strides in all three joints are almost the same.

For 25 strides the variation in random values, total time, and the number of iterations has a zigzag pattern, this shows that they do not reach some stable range of value.

For 50 strides, 75 strides, 100 strides the variation in the values of random values, total time and the number of iterations takes a range starting from a maximum of 51-52 to a minimum of 21-23. In most cases the number of iterations comes out to be between 21-23 and the time taken is in milliseconds to complete the iterations. Mean random values usually vary between 0.4 to 0.6 which shows a dilemma between exploration and exploitation. There is some variation in total execution time which includes the time of executing the iteration as well as writing optimal data in the lookup table. As the number of strides increases, table size also increases as well as if some condition has occurred previously then data is not stored again in the lookup table. This is a time-consuming task as first have to search if data is already stored if yes the skip if data does not exist then go to the end of the lookup table and store data.

For 150 strides and 200 strides the learning time is very less but storing data in lookup takes time and so the total execution time of the episodes increases. The values of random values, total time, and the number of iterations take a range starting from a maximum of 51-52 to a minimum of 21-23.

9.3.3 Execution Phase

In the execution phase, execution starts from 25 strides, 50 strides, 75 strides, 100 strides, 150 strides, 200 strides. The trends observed in these strides in all three joints are almost the same. There are many variations in the number of iterations in the execution phase, the bipedal reads the current position of the joint and then searches in the lookup table for the defined angle, the optimal actions, the next probable state, and the optimal policy and reaches the goal state or the subsequent stable state for the corresponding joint.

For 25 strides the variation in random values, total time, and the number of iterations has a zigzag pattern, this shows that they do not reach some stable range of value.

For 50 strides, 75 strides, 100 strides the variation in the values of random values, total time, and the number of iterations takes a range starting from a maximum of 18 to a minimum of 2. In most cases, the time taken is in milliseconds to complete the iterations but the total execution time is considerable, mean random values usually vary between 0.4 to 0.6 which shows a dilemma between exploration and exploitation. There are some variations in total execution time which include the time of executing the iteration as well as searching and reading optimal data from a lookup table.

As the number of strides increases, the time consumed in the task of searching the data in the lookup table is considerably more. This, in turn, increases the execution time which results in the approximately same time for the execution phase as the learning phase.

For 150 strides and 200 strides, the execution time is more as searching data in lookup takes time and so the total execution time of the episodes increases. The values of random values, total time, and the number of iterations take a range starting from a maximum of 18 to a minimum of 2.

9.4 Value comparison of Hip, Knee and Ankle Joints in Learning and Execution Phase

For Hip Joint (Sharma et al., 2020)

In the first episode goal of walking stable is achieved in 47 iterations

The starting angle of the hip joint is -45^o

For Knee Joint

In the first episode goal of walking stable is achieved in 51 iterations

The starting angle of the hip joint is 0^o

For Ankle Joint

In the first episode goal of walking stable is achieved in 51 iterations

The starting angle of the hip joint is -30^o

9.4.1 Comparison for Number of Iterations in 1st Episode

For Hip, Knee, and Ankle Joint

In the 1st episode goal of walking stable is achieved in 51-43 iterations for each of the joints. Table 9.7, 9.8, 9.9 shows the reduced number of iteration in the 1st episode for each of the joints.

9.4.2 Comparison for Number of Iterations in 200th Episode

For Hip, Knee, and Ankle Joint

In the 200th episode goal of walking stable is achieved in 21-23 iterations for each of the joints. Table 9.10, 9.11, 9.12 shows the reduced number of iteration in the 200th episode for each of the joints.

Table 9.7 Hip Joint (Final Data Episode 1)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
43	0.47244	1.39685	0.10581	0	2.49E-09	2.49E-09	1	0	0
				6.83E-09	0	6.83E-09			
				1.84E-08	0	1.84E-08			
				4.86E-08	4.86E-08	4.86E-08			
				1.32E-07	0	1.32E-07			
				6.69E-07	3.52E-07	6.69E-07			
				0	9.47E-07	9.47E-07			
				0	2.55E-06	2.55E-06			
				0	6.86E-06	6.86E-06			
				1.85E-05	0	1.85E-05			
				0	4.97E-05	4.97E-05			
				0.000136	0	0.000136			
				0.000367	0	0.000367			
				0.000987	0	0.000987			
				0	0.002606	0.002606			
				0.007146	0	0.007146			
				0.019231	0	0.019231			
				0.051756	0	0.051756			
				0.139287	0	0.139287			
				0.699304	0	0.699304			
				0	0	0			

Table 9.8 Knee Joint (Final Data Episode 1)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
51	0.55394	1.4022	0.21696	4.74E-09	2.49E-09	4.74E-09	0	0	0
				6.71E-09	0	6.71E-09			
				1.84E-08	0	1.84E-08			
				4.86E-08	4.86E-08	4.86E-08			
				1.32E-07	0	1.32E-07			
				3.59E-07	0	3.59E-07			
				9.65E-07	0	9.65E-07			
				0	2.55E-06	2.55E-06			
				6.99E-06	0	6.99E-06			
				1.88E-05	0	1.88E-05			
				0	4.97E-05	4.97E-05			
				0.000136	0	0.000136			
				0.000367	0	0.000367			
				0.000987	0	0.000987			
				0.002655	0	0.002655			
				0.007146	0	0.007146			
				0.019231	0	0.019231			
				0.051756	0	0.051756			
				0.139287	0	0.139287			
				0.699304	0	0.699304			
				0	0	0			

Table 9.9 Ankle Joint (Final Data Episode 1)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
51	0.55394	1.4022	0.18134	4.74E-09	2.49E-09	4.74E-09	0	0	0
				6.71E-09	0	6.71E-09			
				1.84E-08	0	1.84E-08			
				4.86E-08	4.86E-08	4.86E-08			
				1.32E-07	0	1.32E-07			
				3.59E-07	0	3.59E-07			
				9.65E-07	0	9.65E-07			
				0	2.55E-06	2.55E-06			
				6.99E-06	0	6.99E-06			
				1.88E-05	0	1.88E-05			
				0	4.97E-05	4.97E-05			
				0.000136	0	0.000136			
				0.000367	0	0.000367			
				0.000987	0	0.000987			
				0.002655	0	0.002655			
				0.007146	0	0.007146			
				0.019231	0	0.019231			
				0.051756	0	0.051756			
				0.139287	0	0.139287			
				0.699304	0	0.699304			
				0	0	0			

Table 9.10 Hip Joint (Final Data Episode 200)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
21	0.49883	0.96286	0.00161	0.483517	0.537241	0.537241	1	1	1
				0.537241	0.596934	0.596934			
				0.596934	0.66326	0.66326			
				0.736226	0.736956	0.736956			
				0.736964	0.81884	0.81884			
				0.909813	0.909822	0.909822			
				1.009902	1.010913	1.010913			
				1.010914	1.123235	1.123235			
				1.246801	1.248036	1.248036			
				1.385326	1.386699	1.386699			
				1.386731	1.540756	1.540756			
				1.540841	1.711896	1.711896			
				1.900058	1.901956	1.901956			
				1.904653	2.112881	2.112881			
				2.116832	2.346559	2.346559			
				2.601829	2.604363	2.604363			
				2.619011	2.885866	2.885866			
				2.918133	3.185337	3.185337			
				3.482238	3.482259	3.482259			
				3.715767	0	3.715767			
				0	0	0			

Table 9.11 Knee Joint (Final Data Episode 200)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
21	0.54280	0.96286	0.00615	0.483516	0.537241	0.537241	1	1	1
				0.537241	0.596934	0.596934			
				0.59694	0.66326	0.66326			
				0.66399	0.736956	0.736956			
				0.736955	0.81884	0.81884			
				0.81974	0.909822	0.909822			
				0.910823	1.010912	1.010912			
				1.010914	1.123235	1.123235			
				1.123252	1.248036	1.248036			
				1.249421	1.386699	1.386699			
				1.386731	1.540756	1.540756			
				1.710202	1.711895	1.711895			
				1.714003	1.901956	1.901956			
				2.110799	2.112881	2.112881			
				2.114535	2.346559	2.346559			
				2.604338	2.604363	2.604363			
				2.619011	2.885866	2.885866			
				3.182664	3.185337	3.185337			
				3.272123	3.482259	3.482259			
				3.715767	0	3.715767			
				0	0	0			

Table 9.12 Ankle Joint (Final Data Episode 200)

Iterations	Mean Random	Total Reward	Total Time	Q(:,1)	Q(:,2)	C	Action 1	Action 2	Action 3
23	0.52326	0.23914	0.00097	8.55E-09	1.17E-05	1.17E-05	1	1	1
				4.87E-06	3.11E-05	3.11E-05			
				3.1E-05	8.46E-05	8.46E-05			
				1.98E-07	0.000229	0.000229			
				0.000256	0.000611	0.000611			
				0.000286	0.001645	0.001645			
				0.000687	0.004444	0.004444			
				0.001832	0.011983	0.011983			
				0.011947	0.032218	0.032218			
				0.002313	0.086632	0.086632			
				0.014933	0.348896	0.348896			
				0.629748	0.390436	0.629748			
				0.634336	0.93425	0.93425			
				0.000987	1.231763	1.231763			
				0.007109	1.468888	1.468888			
				1.521333	1.733414	1.733414			
				1.748164	2.1041	2.1041			
				1.960922	2.323825	2.323825			
				2.204555	2.525091	2.525091			
				2.652246	0	2.652246			
				0	0	0			

9.4.3 Comparison of Reduction in Number of Iterations in Successive Episodes For Hip, Knee, and Ankle Joint

In all three joints number of iterations reduces in the execution phase as compared to previous learning phase episodes. After a point of time, it stays in between a specific range from 21 - 25 iterations per episode for the learning phase. This shows that the bipedal is learning from its experience of previous episodes and using that in learning further(Sharma et al., 2020).

These values are stored for a run as execute, the algorithm for next gait of the bipedal process all values are reset, and it restarts its learning as the environment is dynamic. After a few episodes, the bipedal starts using the optimal actions from the action set as they are stored in the lookup table. The values are retrieved and the scenarios are compared if the position of an object in the coordinate system is the same then the dynamics of the system are known to the bipedal. The bipedal exploits the previous knowledge data which it has learned in the learning phase and trains fast in the execution phase and reached the soccer ball.

If the scenario changes mean the dynamics of the environment are changing then the RL agent executes the proposed learning algorithm from scratch. The execution of the proposed forgetting mechanism RL algorithm depends on the execution of the RL based object identification algorithm whose result is the dynamics of the environment. The dynamics of the environment are calculated then compares if the same scenario exists in the previously learned knowledge, then executes that same sequence of steps that were successful and gave optimal policy to follow and stored in the lookup table. If such a scenario does not exist means no such data is found in the lookup table then bipedal run in the dynamics of the environment and store the successful and exploited state-action pair in the state action lookup table.

Table 9.13 Comparison of Reduction in Number of Iterations in Successive Episodes

HIP JOINT		KNEE JOINT		ANKLE JOINT	
Episodes	**Iterations**	**Episodes**	**Iterations**	**Episodes**	**Iterations**
1	43	1	49	1	51
2	37	2	45	2	35
3	29	3	41	3	31
4	33	4	43	4	31
5	33	5	37	5	29
6	29	6	30	6	29
7	31	7	25	7	27
8	29	8	27	8	27
9	25	9	25	9	25
10	23	10	23	10	23
11	21	11	23	11	22
12	25	12	23	12	23
13	27	13	21	13	22
14	27	14	21	14	23
15	25	15	25	15	22
16	23	16	23	16	23
17	21	17	22	17	25
18	23	18	22	18	25
19	23	19	21	19	25
20	23	20	23	20	27
21	24	21	22	21	23
22	26	22	23	22	25
23	25	23	23	23	27
24	22	24	21	24	25

9.4.4 Comparison of Number of Iterations Vs Episodes in Learning and Execution phase for all joints

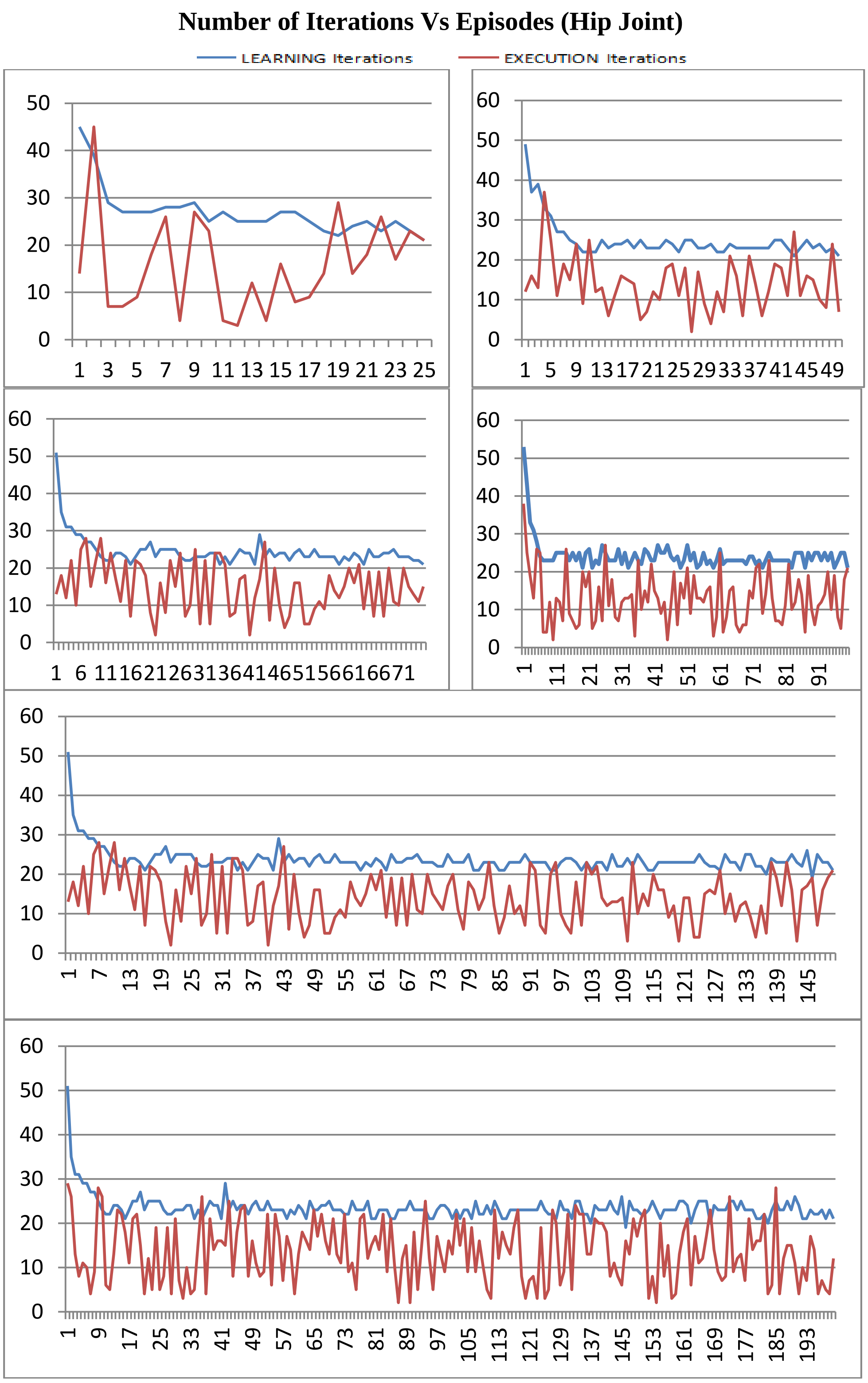


Figure 9.13 Comparison of Hip for 25, 50, 75, 100, 150, 200 strides

Figure 9.13 shows that in the learning phase as strides increases from 25 to 200 number of minimum iterations required to learn decreases from 51 to 21. This reveals that the bipedal is using the lookup data when the dynamics of the system is not changing. In the executing phase as strides increases from 25 to 200 number of minimum iterations required to execute varies from 2 to 18 depending on the start angle of the hip joint.

As the knee is attached with the pelvis whose center is usually COM of the biped. To learns the stable position of the hip joint is relatively easy as compared to the knee and ankle joint. Biped to be stable has some approximate angle so that the COM of the biped is in between both legs.

Figure 9.14 shows that in the learning phase as strides increases from 25 to 200 number of minimum iterations required to learn decreases from 47 to 21. This reveals that the bipedal is using the lookup data when the dynamics of the system is not changing. In the executing phase as strides increases from 25 to 200 number of minimum iterations required to execute varies from 2 to 18 depending on the start angle of the knee joint. The knee joint angle varies as the gait proceeds, but to reach an angle at which the knee joint is stable and the bipedal does not get tumbled or imbalanced the current joint angle value plays an important role. After getting the current state and knows the goal state (approximately) different actions have opted means different options of how the knee joint should reach a stable angle.

Number of Iterations Vs Episodes (Knee Joint)

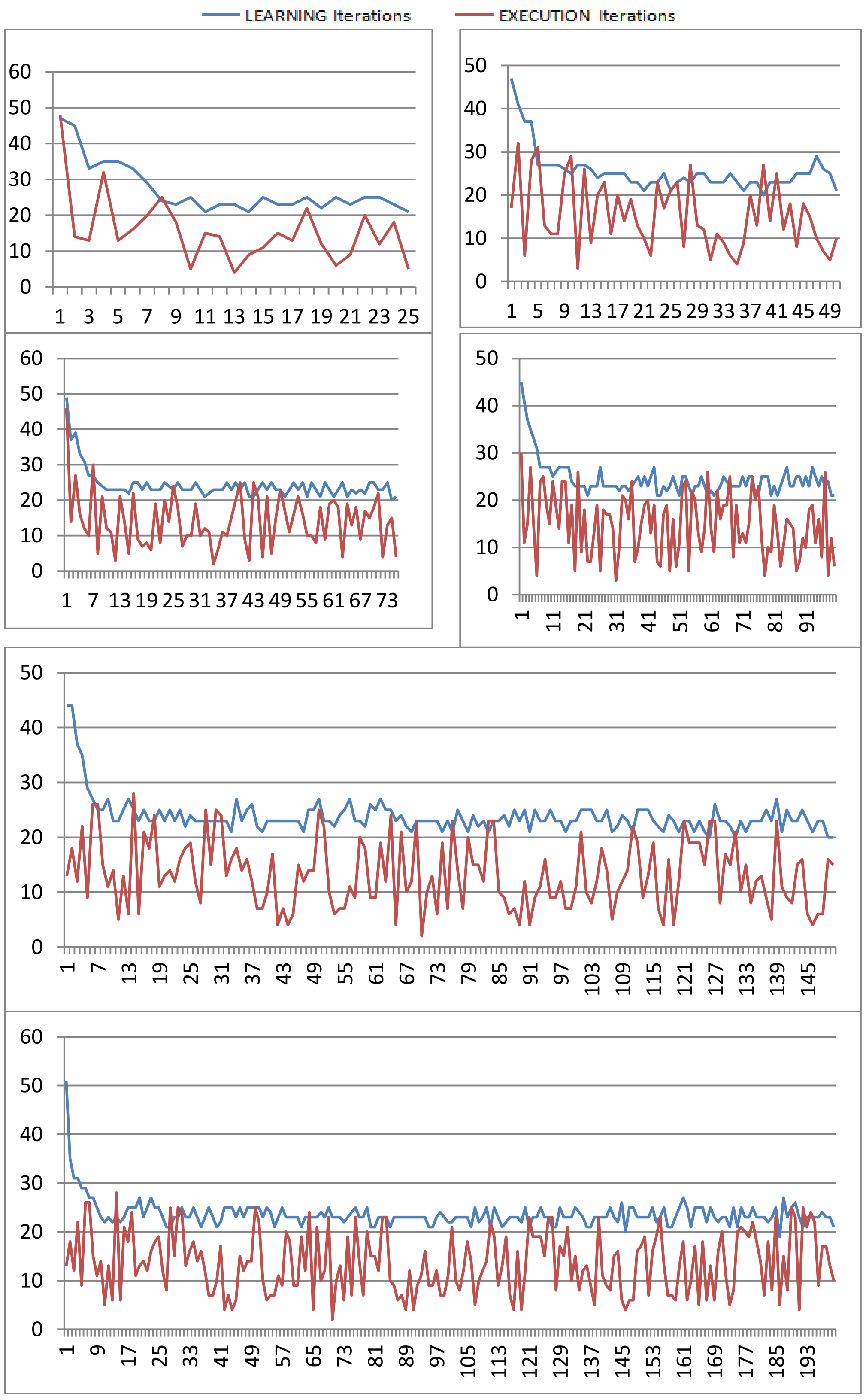


Figure 9.14 Comparison of Knee for 25, 50, 75, 100, 150, 200 strides

Figure 9.15 shows that in the learning phase as strides increases from 25 to 200 number of minimum iterations required to learn decreases from 51 to 21. This reveals that the bipedal is using the lookup data when the dynamics of the system is not changing. In the executing phase as strides increases from 25 to 200 number of minimum iterations required to execute varies from 2 to 18 depending on the start angle of the ankle joint. The most difficult joint to train is the ankle joint.

The system follows the hierarchical structure first to train the hip joint to reach a stable position than train the knee joint to reach a stable position so that bipedal remains stable. Last but not least ankle joint is trained which has to control the damping and ZMP of the bipedal so has to adjust the ankle joint of the bipedal taking them into account. The ankle joint is calculated considering that the sole is in contact with the ground. When the sole touches the ground the damping comes into the picture as there will be jerks when the sole touches the ground and the angle of the ankle is adjusted in real-time to keep the bipedal stable. This stability is checked by the calculation of ZMP which should lie in the convex hull of the gait.

Figures 9.13, 9.14, 9.15 show a remarkable decrease in the number of iterations in the execution phase of the hip, knee, and ankle joint as compared to the number of iterations in the learning phase of the bipedal.

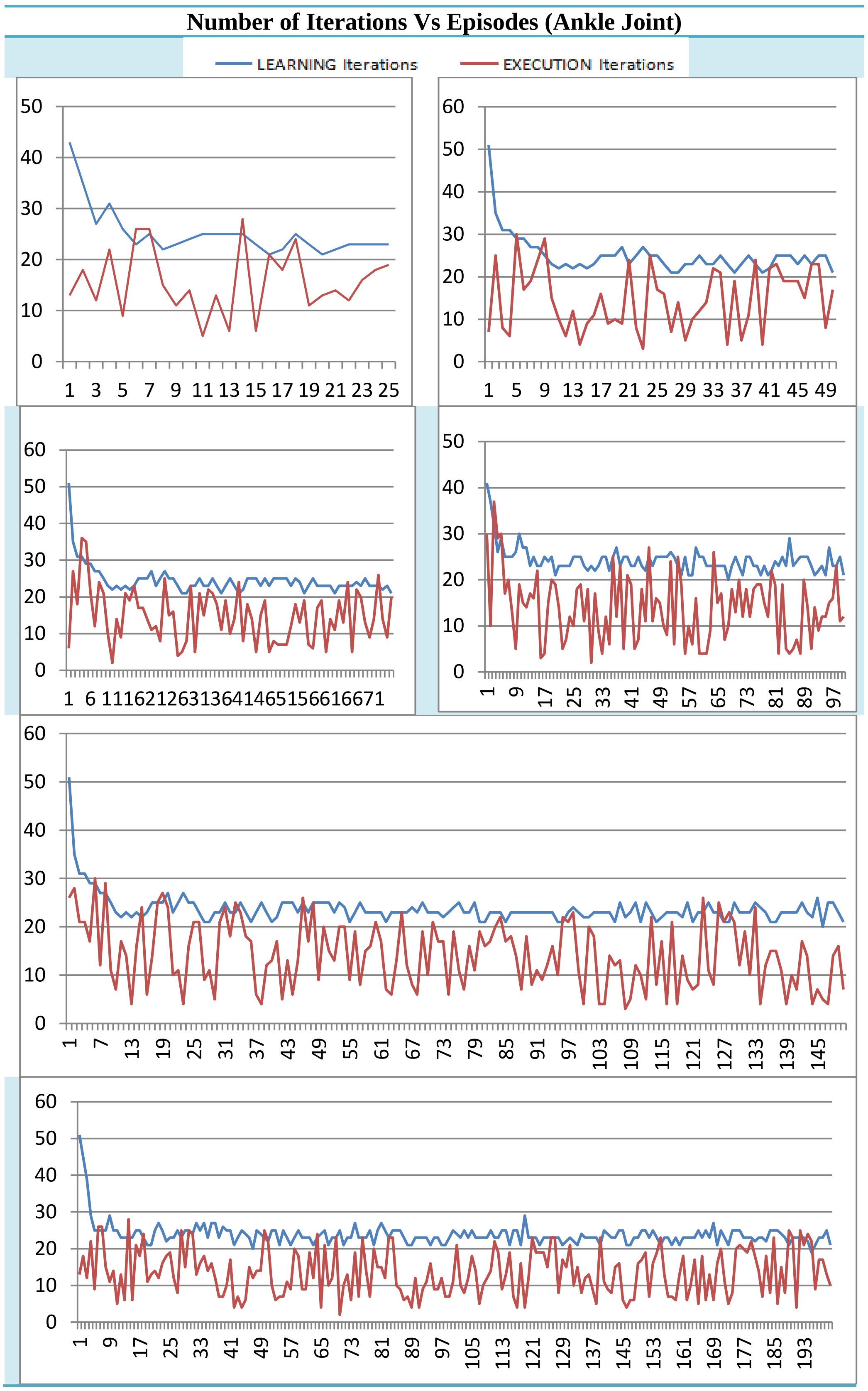


Figure 9.15 Comparison of Ankle for 25, 50, 75, 100, 150, 200 strides

9.4.5 Comparison of Total Time for Learning and Execution phase for Hip, Knee, and Ankle Joint

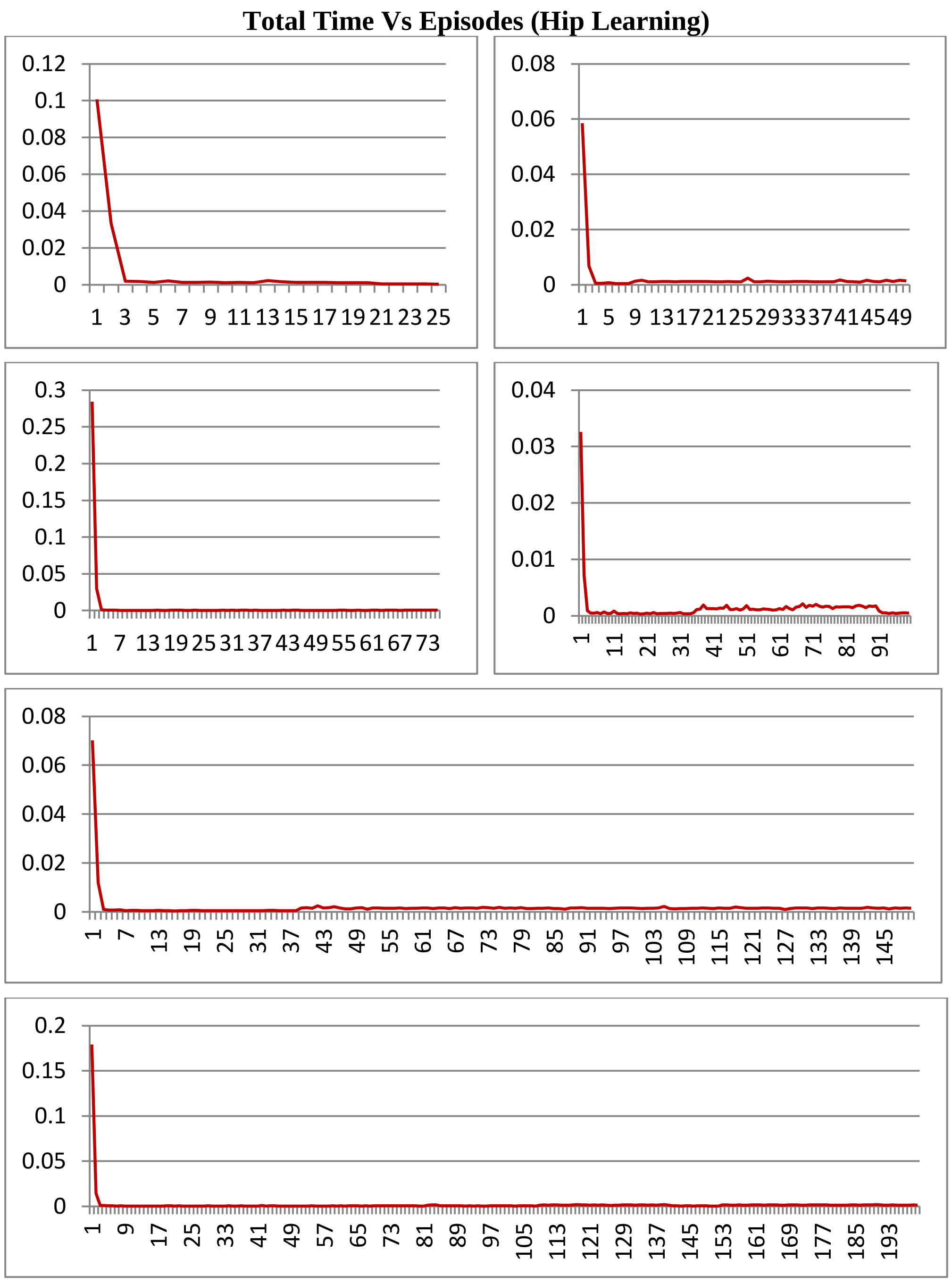


Figure 9.16 Total Time Vs episodes for Hip Learning

As seen in figure 9.16, for hip joint learning there are variations in the hip joint learning as the number of strides/episodes increases. These are visible in 100, 150 strides before, and after that, they have a straight-line graph.

Total Time Vs Episodes (Hip Execution)

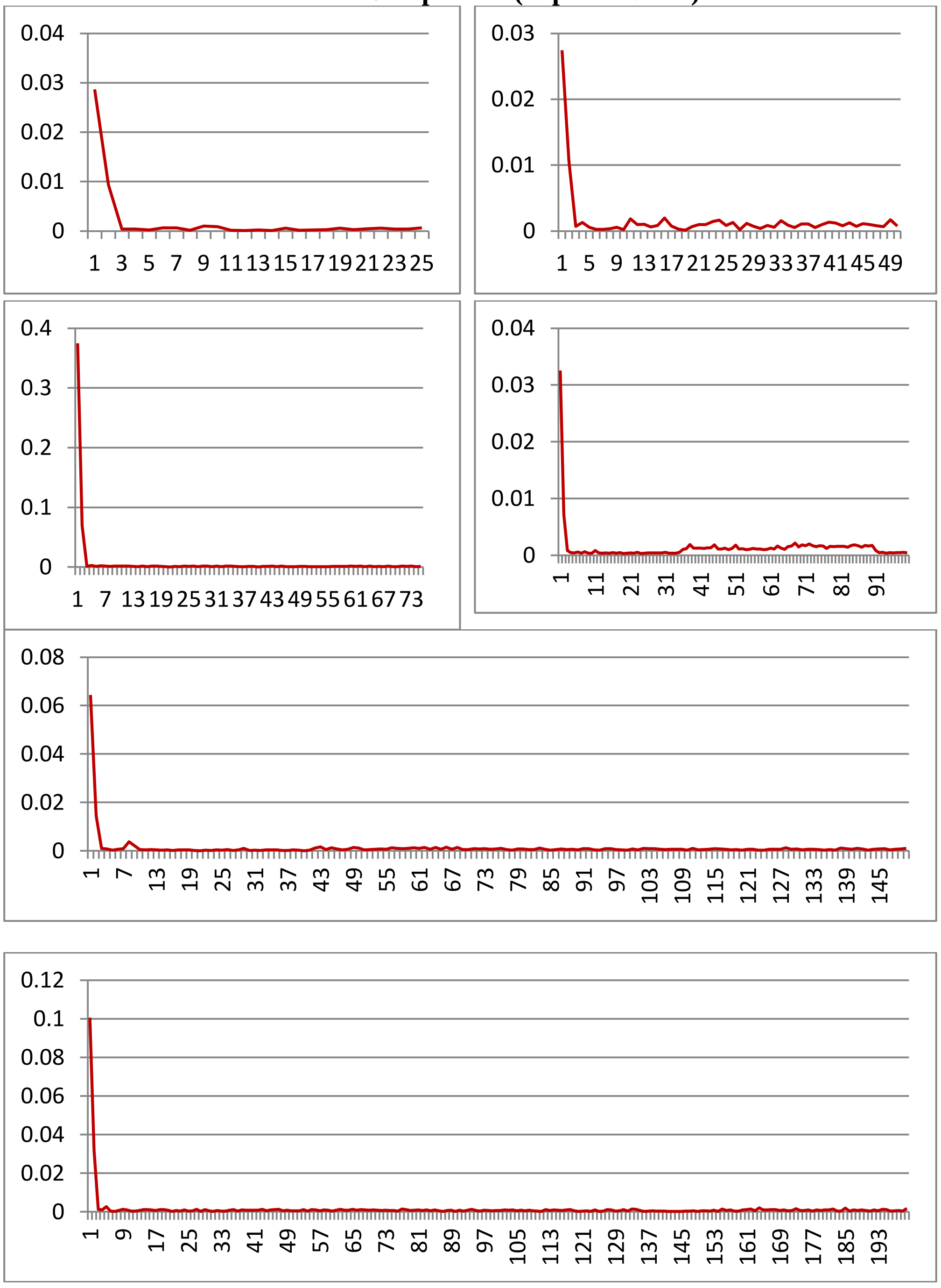


Figure 9.17 Total Time Vs episodes for Hip Execution

As seen in Figure 9.17, the variations are more in the execution phase as compared to the learning phase. In the execution phase, the variations are visible from 50 strides and followed in 75, 100, 150 strides. A stable graph is visible for 25 and 200 episodes.

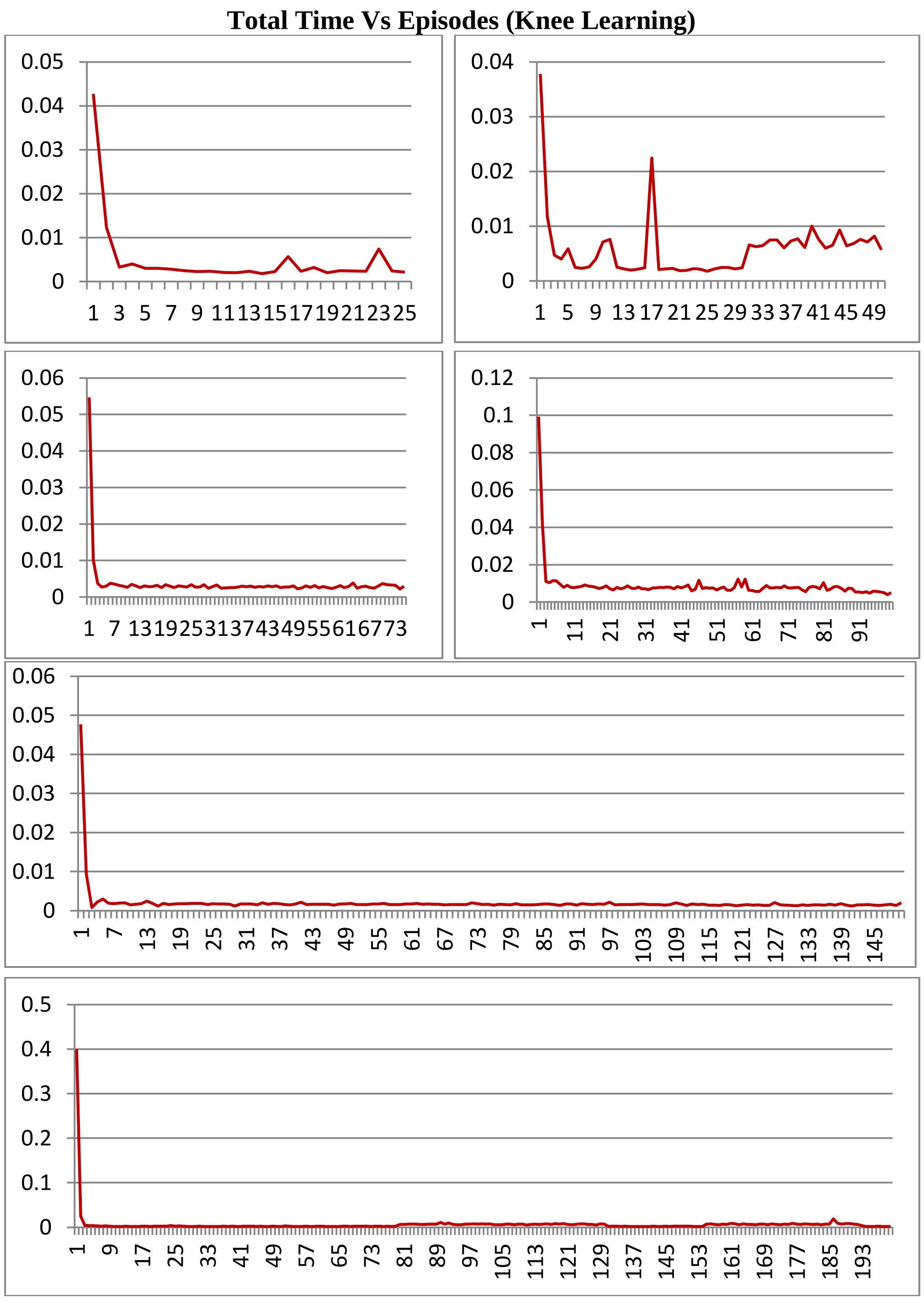


Figure 9.18 Total Time Vs episodes for Knee Learning

As the learning of the joints is done hierarchically first the stable angle of the hip is fixed then the knee joint is learning. The variations of hip joint learning are propagated in the learning of the knee joint. As seen in figure 9.18, the variations are more as compared to the hip joint in all the strides of learning of the knee joint. There are large deviations in the values of the total execution time for almost all strides that are visible in the graph.

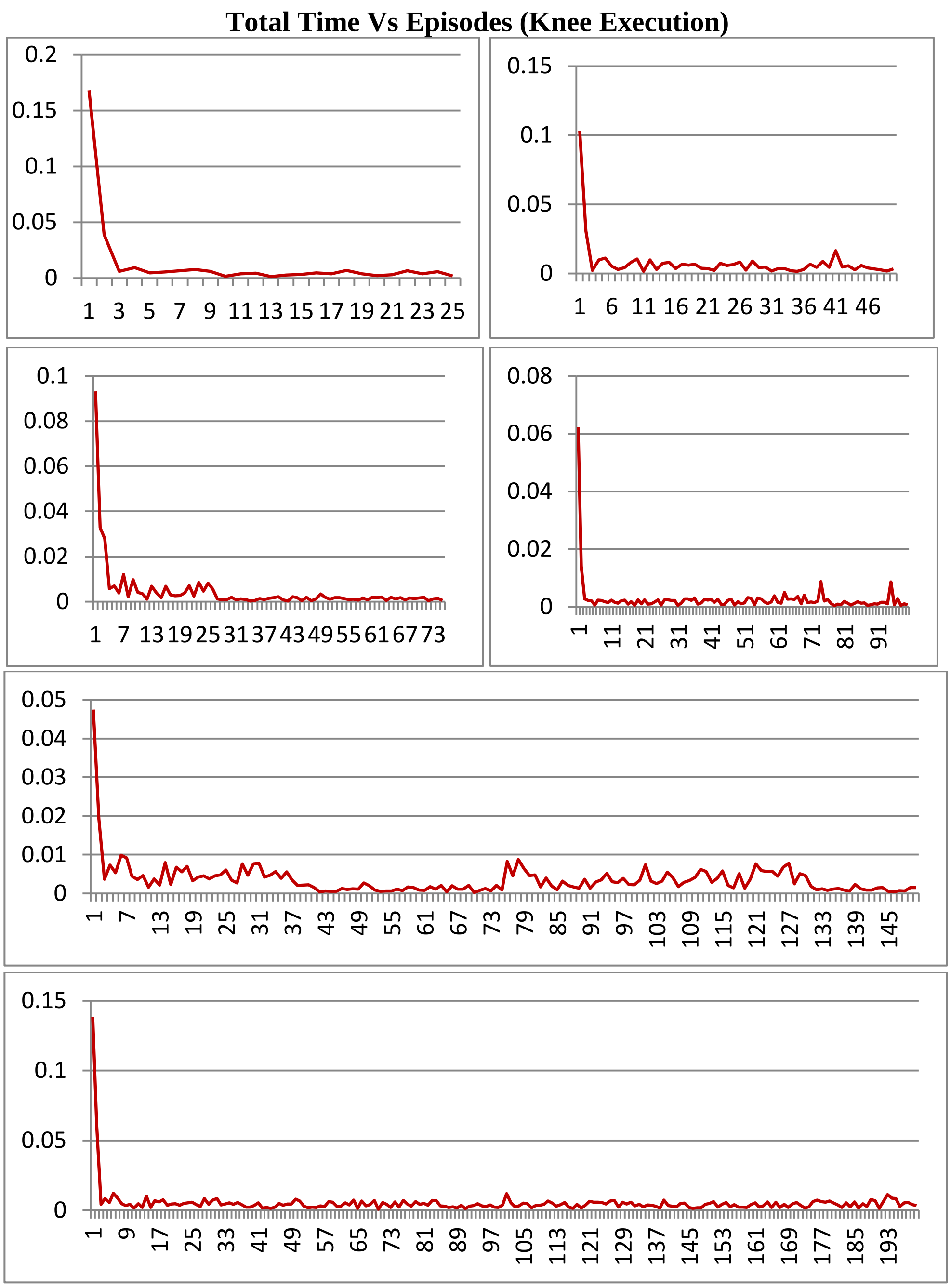


Figure 9.19 Total Time Vs episodes for Knee Execution

As the executing of the joints is also carried out the hierarchically first stable angle of the hip is fixed then the knee joint is executed. The variations of the hip joint executions are propagated in the execution of the knee joint. As seen in figure 9.19, the variations are more as compared to the hip joint in all strides of execution of the knee joint. There are large deviations in the values of the total execution time for almost all strides that are visible in the graph.

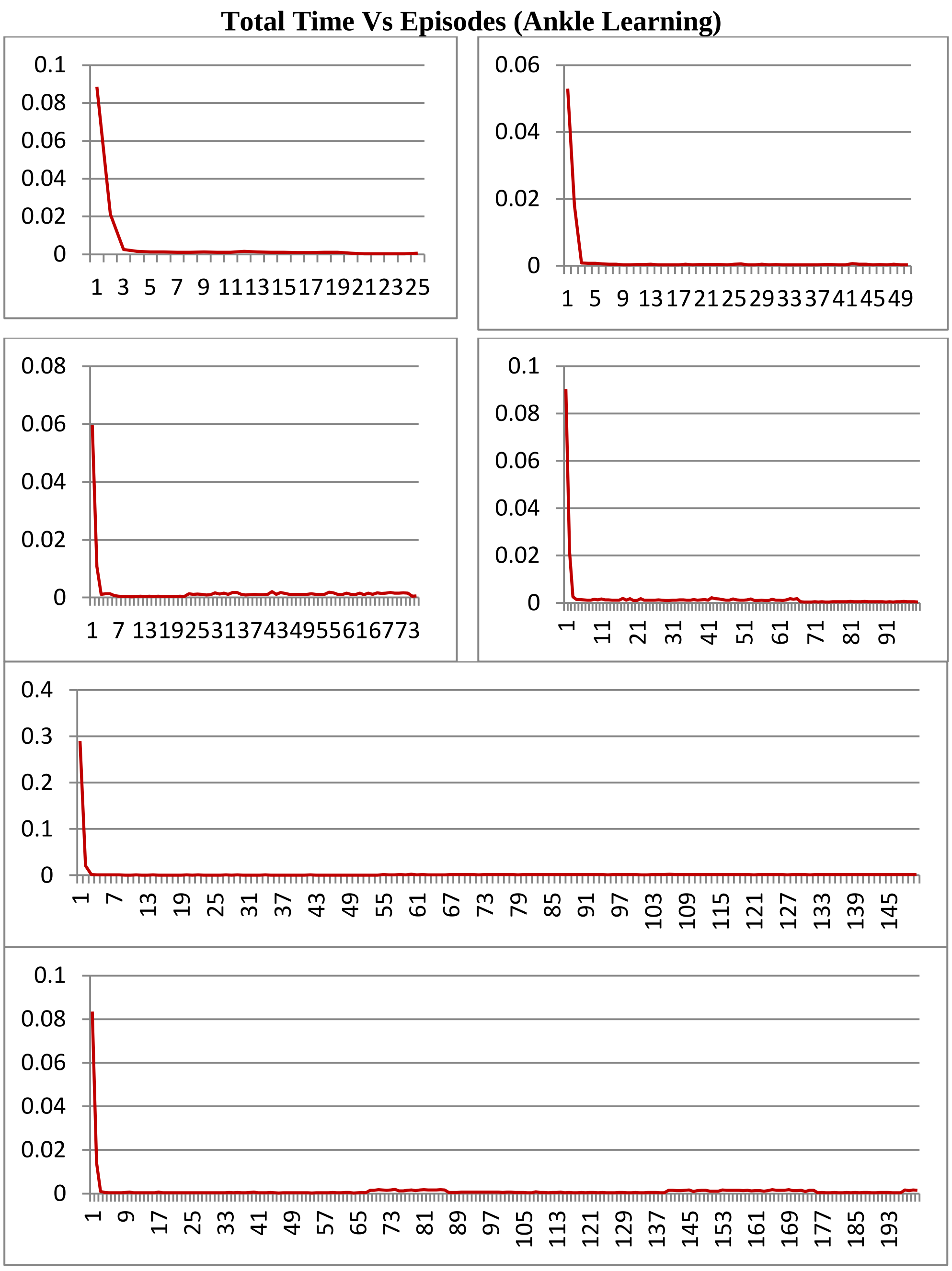


Figure 9.20 Total Time Vs episodes for Ankle Learning

As seen in figure 9.20, the learning phase is far stable as compared to that of hip joint and knee joint as ankle joint also have contact forces to adjust and the zmp compensator at the runtime to stabilize ankle joint angle.

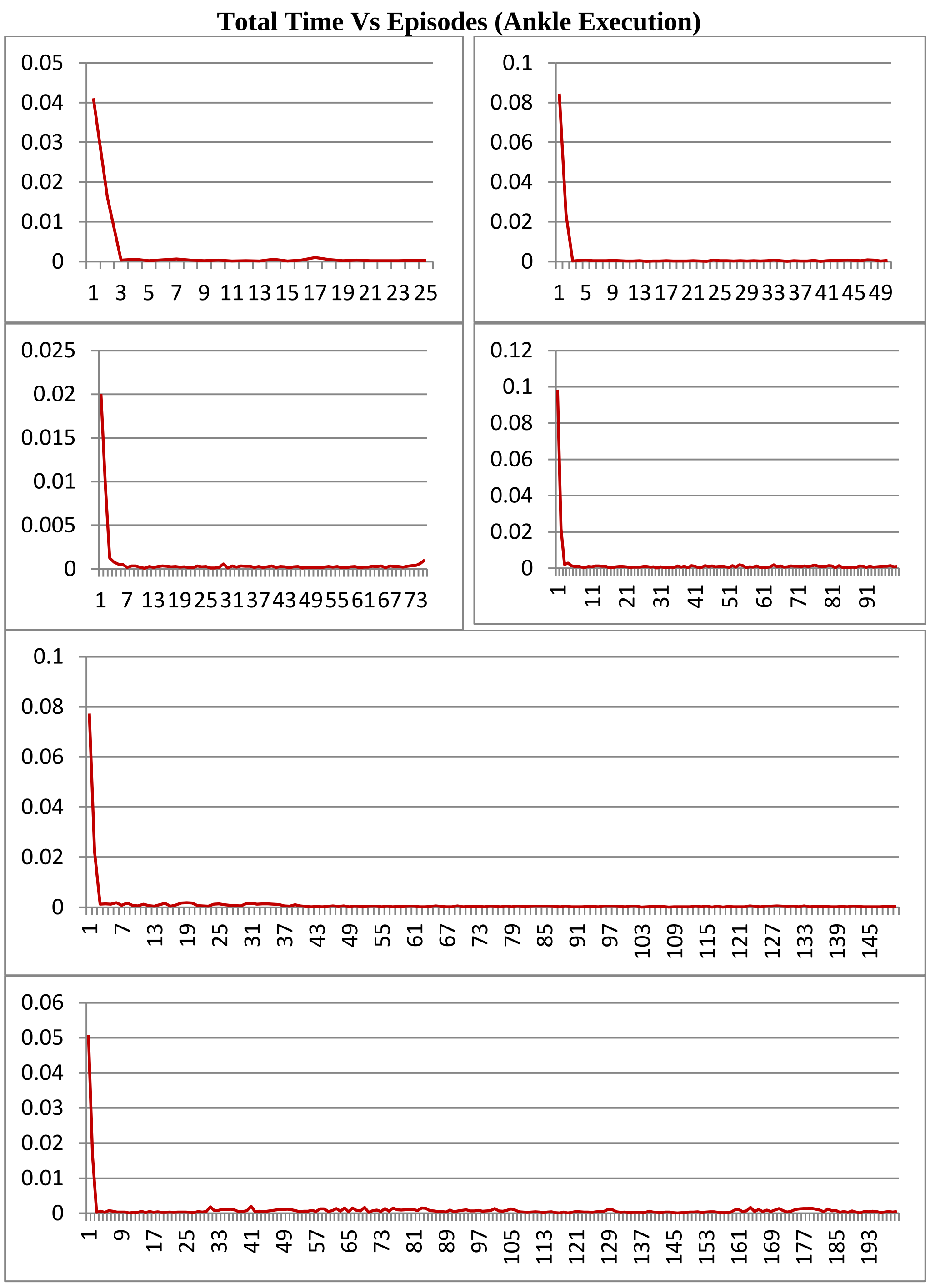


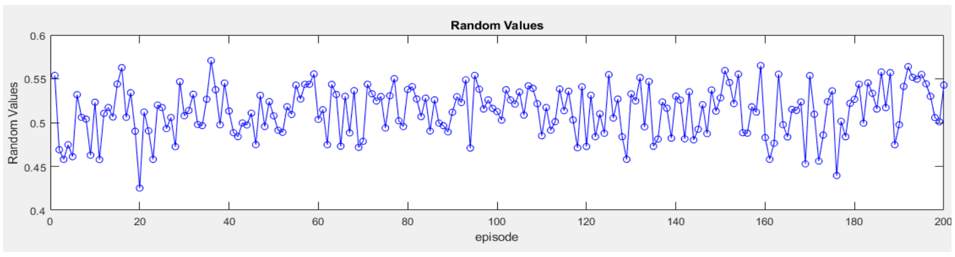
Figure 9.21 Total Time Vs episodes for Ankle Execution

As seen in figure 9.21 there is the least variation in the execution phase as zmp compensator and the contact forces and torques are acting at run time. If the leg is not in contact with the ground then variation in angle is more and the leg should not hit the ground with force to avoid damages in bipedal.

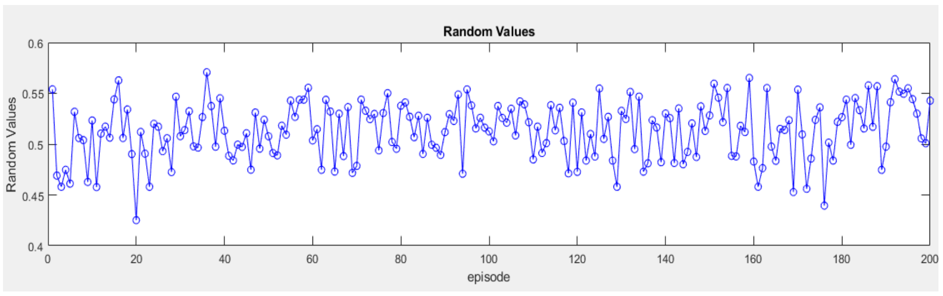
9.4.6 Random Values Generations for 200 Episodes

Action selection depends on the generation of random values. If random value generated < 0.5 then random actions are selected from a defined action set. Hence, bipedal explores the dynamic environment.

Hip Joint



Knee Joint



Ankle Joint

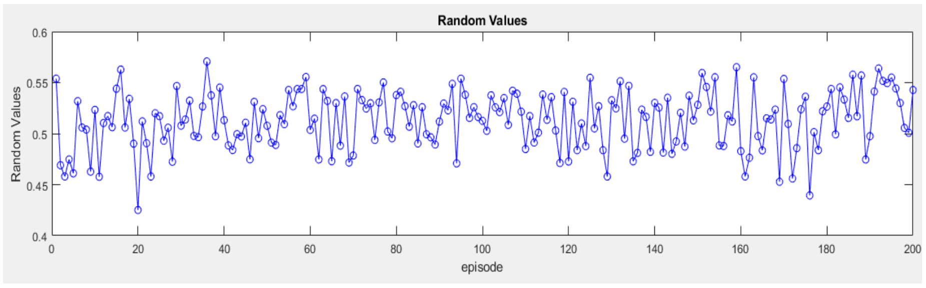


Figure 9.22 Comparison of Random Value Generation for all Three Joints

If the random value generated is greater than 0.5 then greedy (optimal) action is selected from the defined action set. Greedy or optimal actions are actions that are chosen frequently by the bipedal. Hence, bipedal exploits a dynamic environment. This helps bipedal to learn fast and reach the optimal policy with maximum immediate reward. Due to this bipedal has a smooth and stable trajectory without jerks.

9.4.7 Mean Random Values, Total Rewards calculation of Learning and Execution Phase for Hip Joint, Knee Joint, Ankle Joint

Mean Random Value, Total Rewards Vs Episodes (Hip Joint Learning)

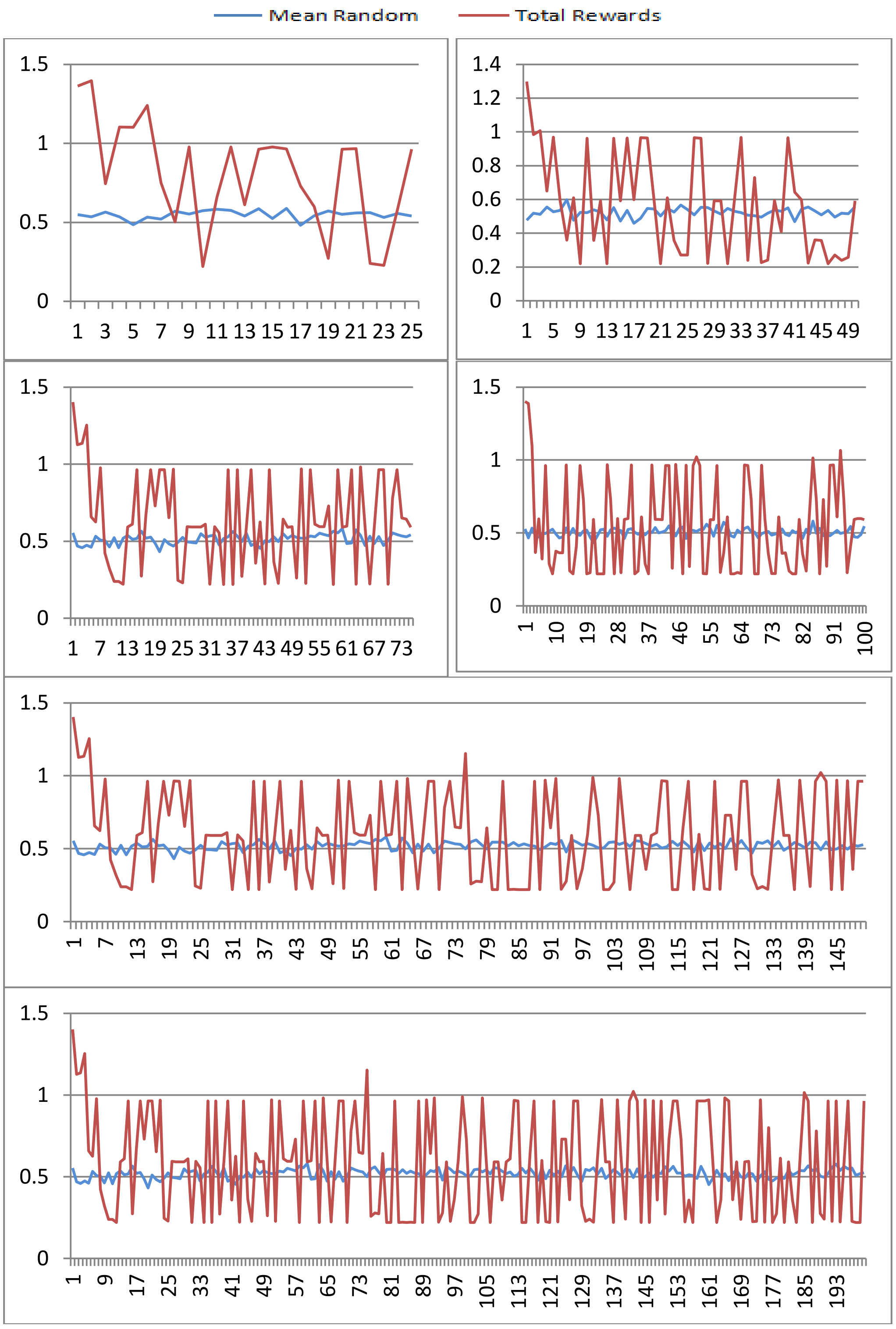


Figure 9.23 Comparison of Learning Phase of Reward Generation of Hip Joint

Figure 9.23 shows that in the learning phase as strides increases from 25 to 200 randomness in rewards generation increases which reveals that bipedal if exploring more optimal actions to reach the goal but, in end, uses greedy actions i.e. exploits greedy action.

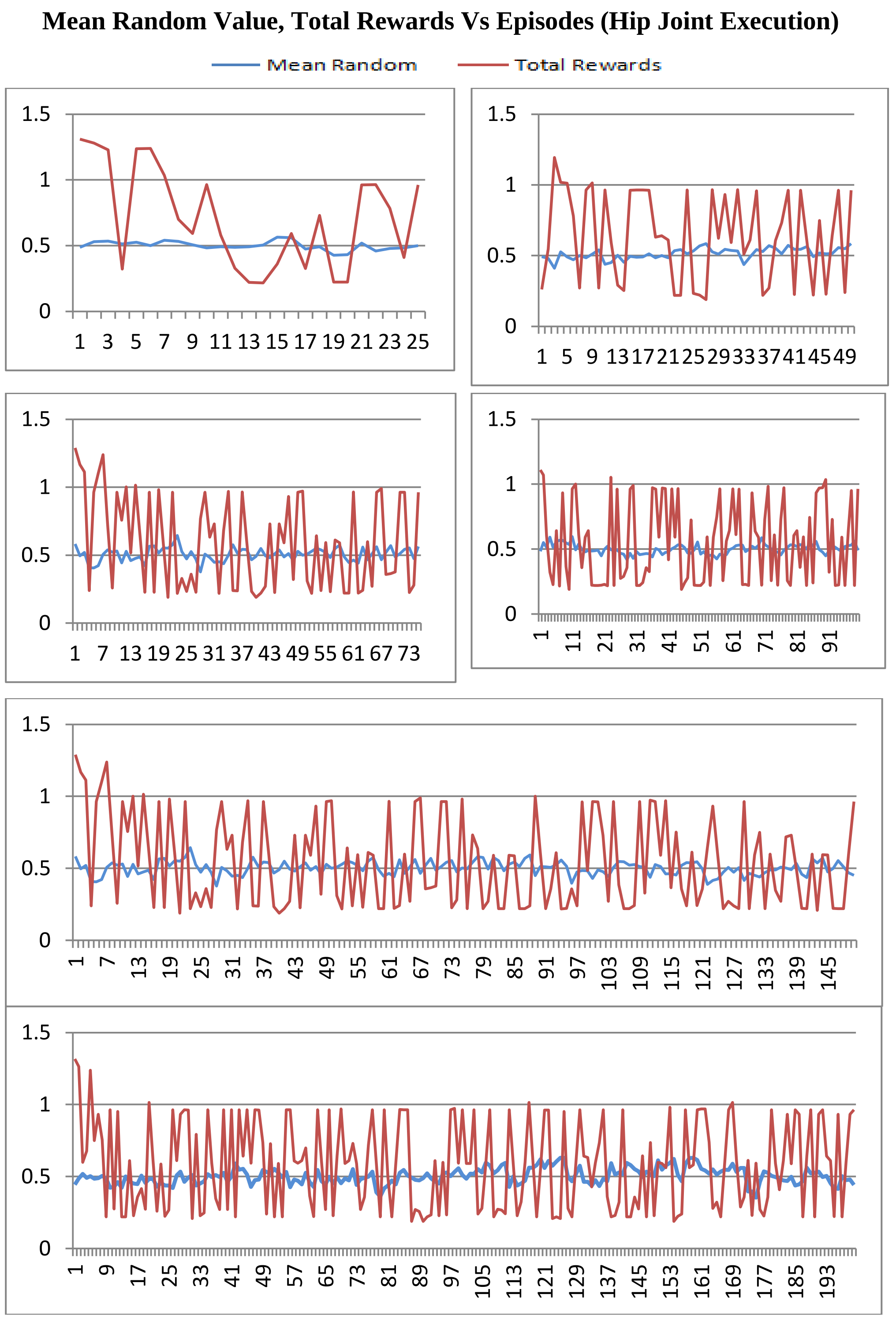


Figure 9.24 Comparison of Executing Phase of Reward Generation of Hip Joint

Figure 9.24 shows that in execution phase as strides increases from 25 to 200 randomness in rewards generation is almost constant which reveals that bipedal if exploiting optimal actions to reach the goal but there are some spikes which are exception or error which shows that hip joint goes to near about its original start position and so randomness in rewards is there.

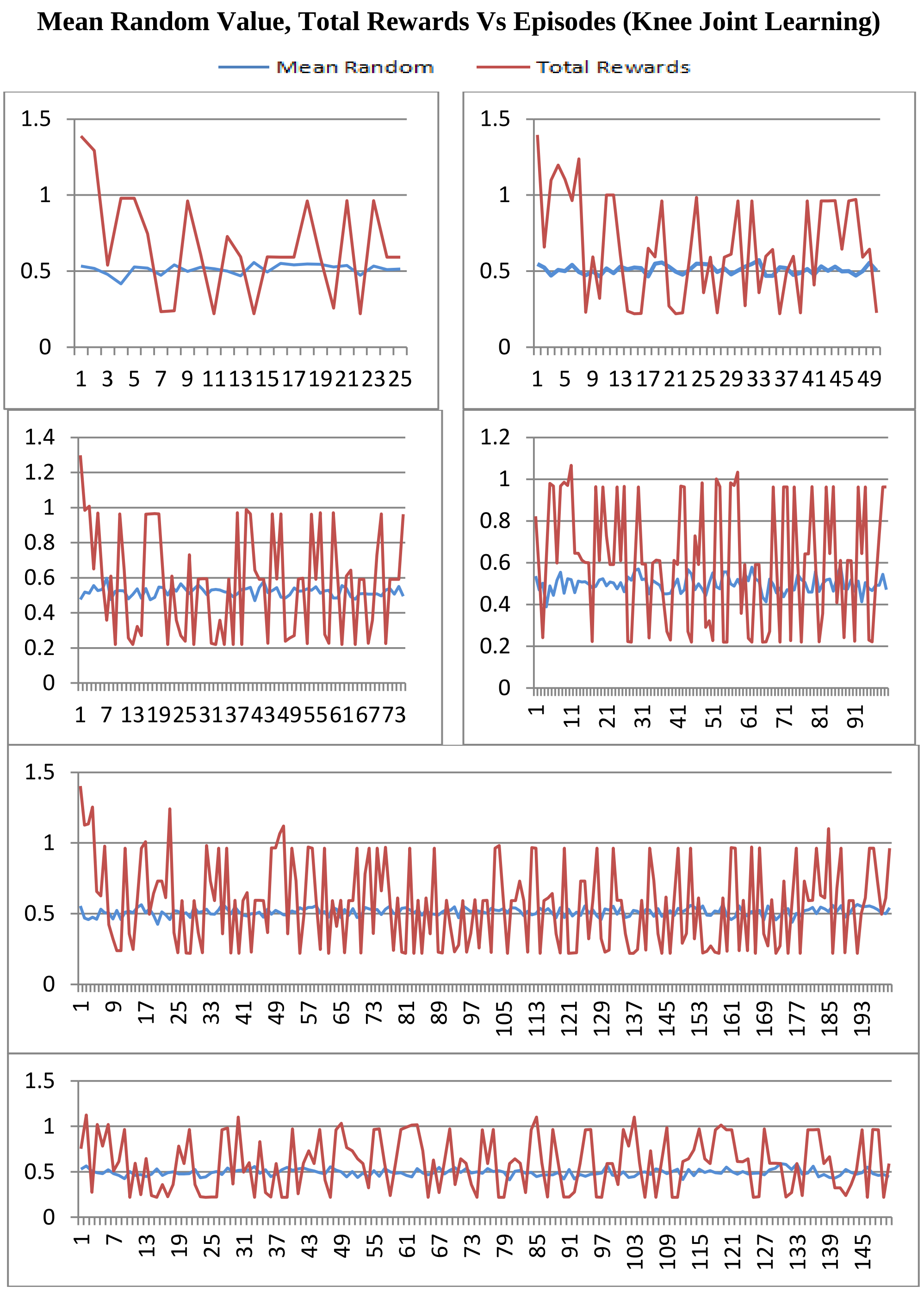


Figure 9.25 Comparison of Learning Phase of Reward Generation of Knee Joint

Figure 9.25 shows that in the learning phase as strides increases from 25 to 200 randomness in rewards generation increases which reveals that bipedal if exploring more optimal actions to reach the goal but, in end, uses greedy actions i.e. exploits greedy action.

Mean Random Value, Total Rewards Vs Episodes (Knee Joint Execution)

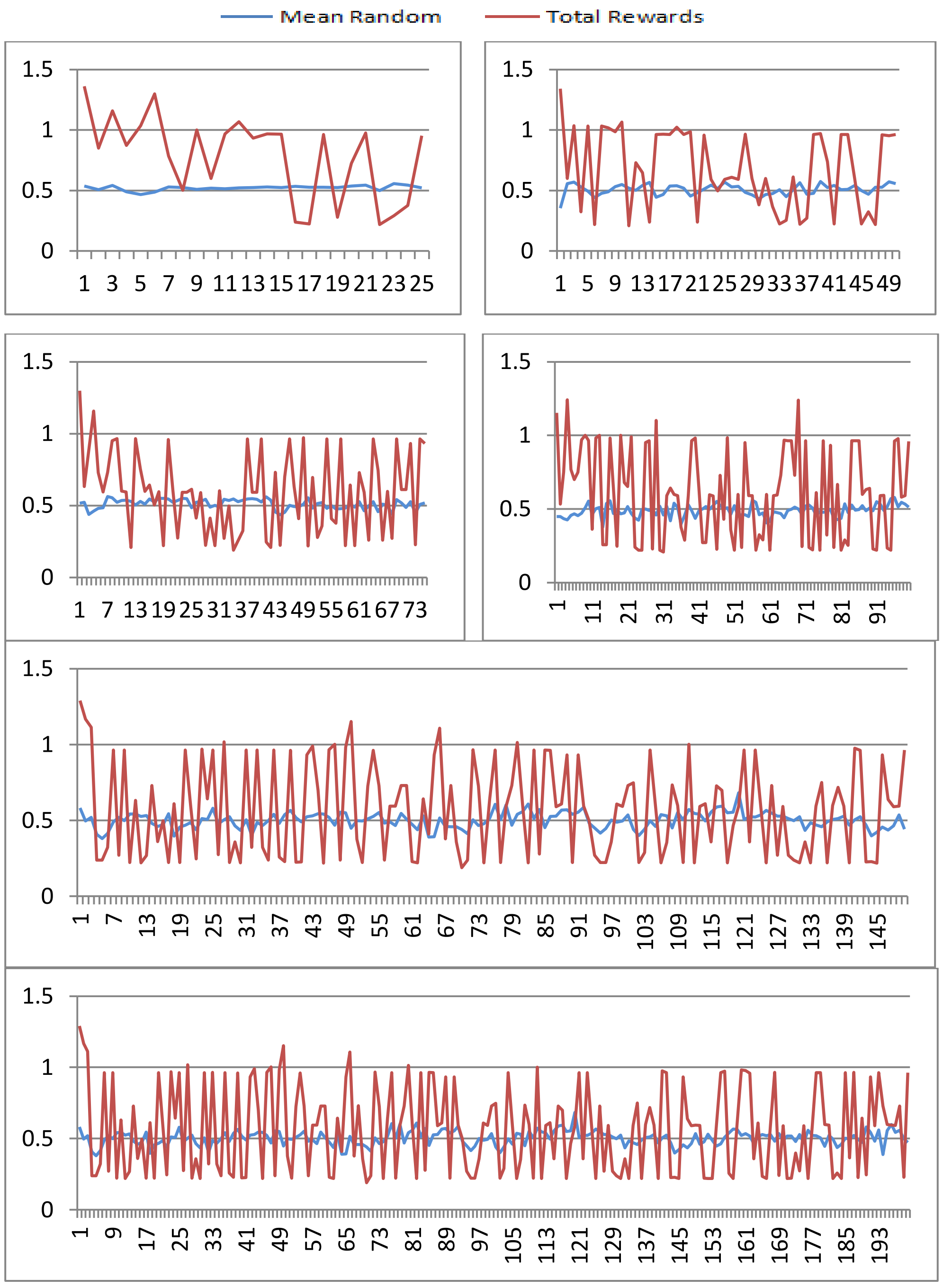
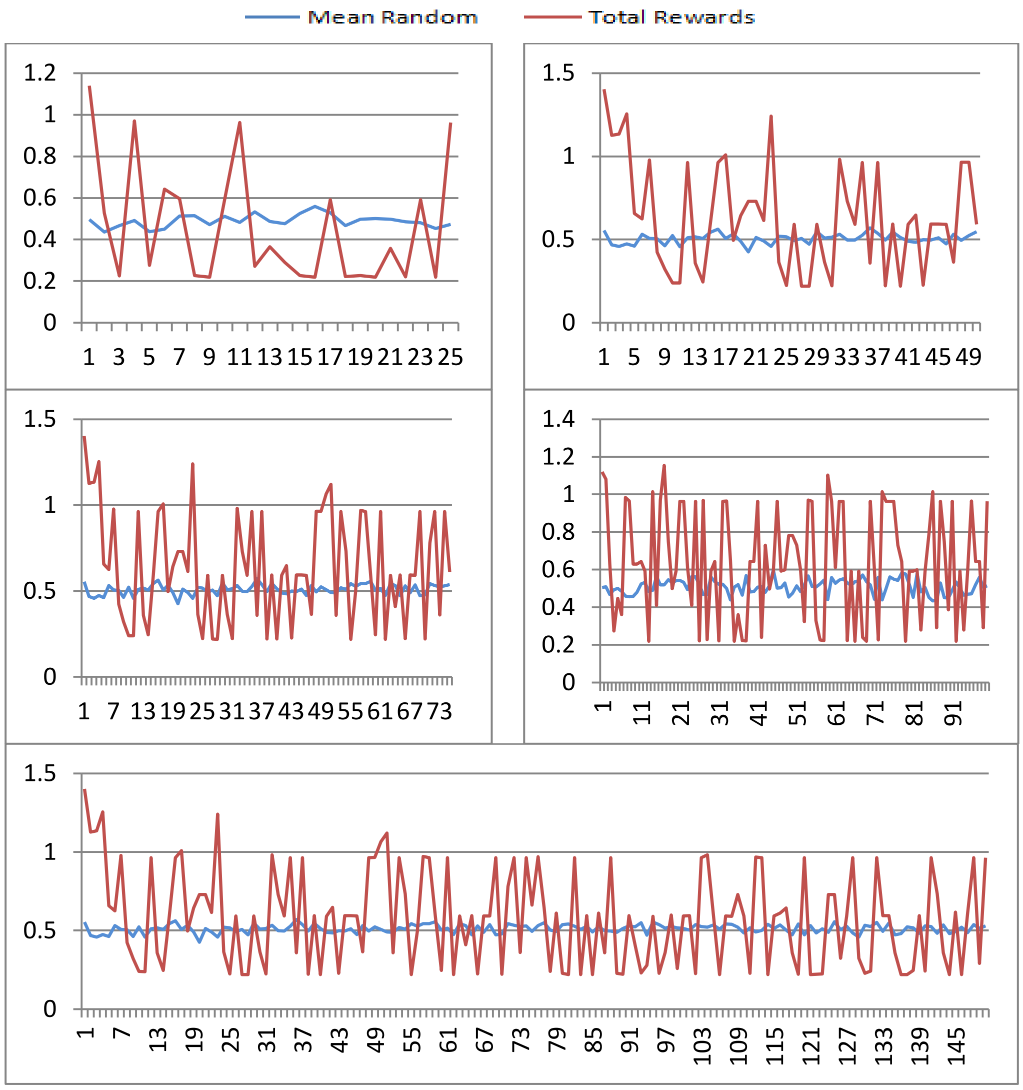


Figure 9.26 Comparison of Executing Phase of Reward Generation of Knee Joint

Figure 9.26 shows that in execution phase as strides increases from 25 to 200 randomness in rewards generation is almost constant which reveals that bipedal if exploiting optimal actions to reach the goal but there are some spikes which are exception or error which shows that knee joint goes to near about its original start position and so randomness in rewards is there. The randomness in the knee is more as compared to the hip joint as joints are trained hierarchically if randomness is there at the hip joint in that stride then it is propagated to the knee joint. More spikes in the graph can be seen of 200 steps.

Mean Random Value, Total Rewards Vs Episodes (Ankle Joint Learning)



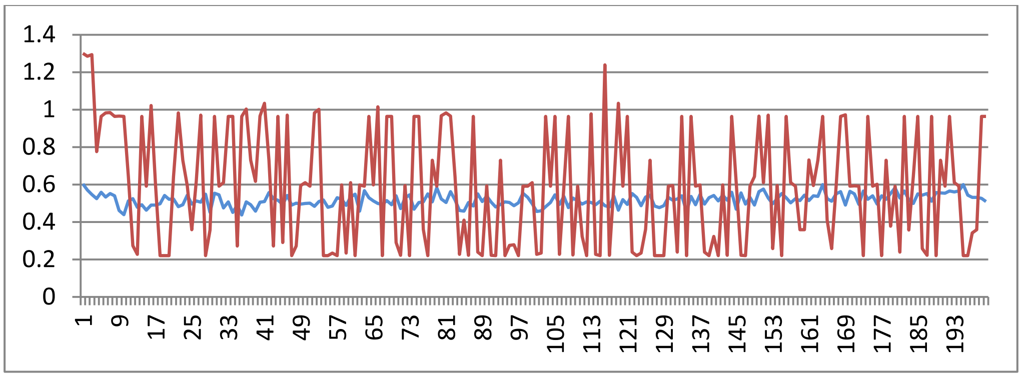


Figure 9.27 Comparison of Learning Phase of Reward Generation of Ankle Joint

Figure 9.27 shows that in the learning phase as strides increases from 25 to 200 randomness in rewards generation increases which reveals that bipedal if exploring more optimal actions to reach the goal but, in end, uses greedy actions i.e. exploits greedy action.

Figure 9.28 shows that in execution phase as strides increases from 25 to 100 randomness in rewards generation is almost constant which reveals that bipedal if exploiting optimal actions to reach the goal but there are some spikes which are exception or error which shows that ankle joint goes to near about its original start position and so randomness in the rewards is there. The randomness of the hip and knee is carried out to the ankle joint as joints are trained hierarchically. If randomness is there at hip and knee joints in that stride, then it is propagated to the ankle joint. But the ankle joint has a damping controller and ZMP controller which helps in minimizing this propagated error so that the stability of the bipedal is not affected.

Bipedal is more stable due to the execution of these controllers in real-time, resulting in fewer spikes in the graph of different strides.

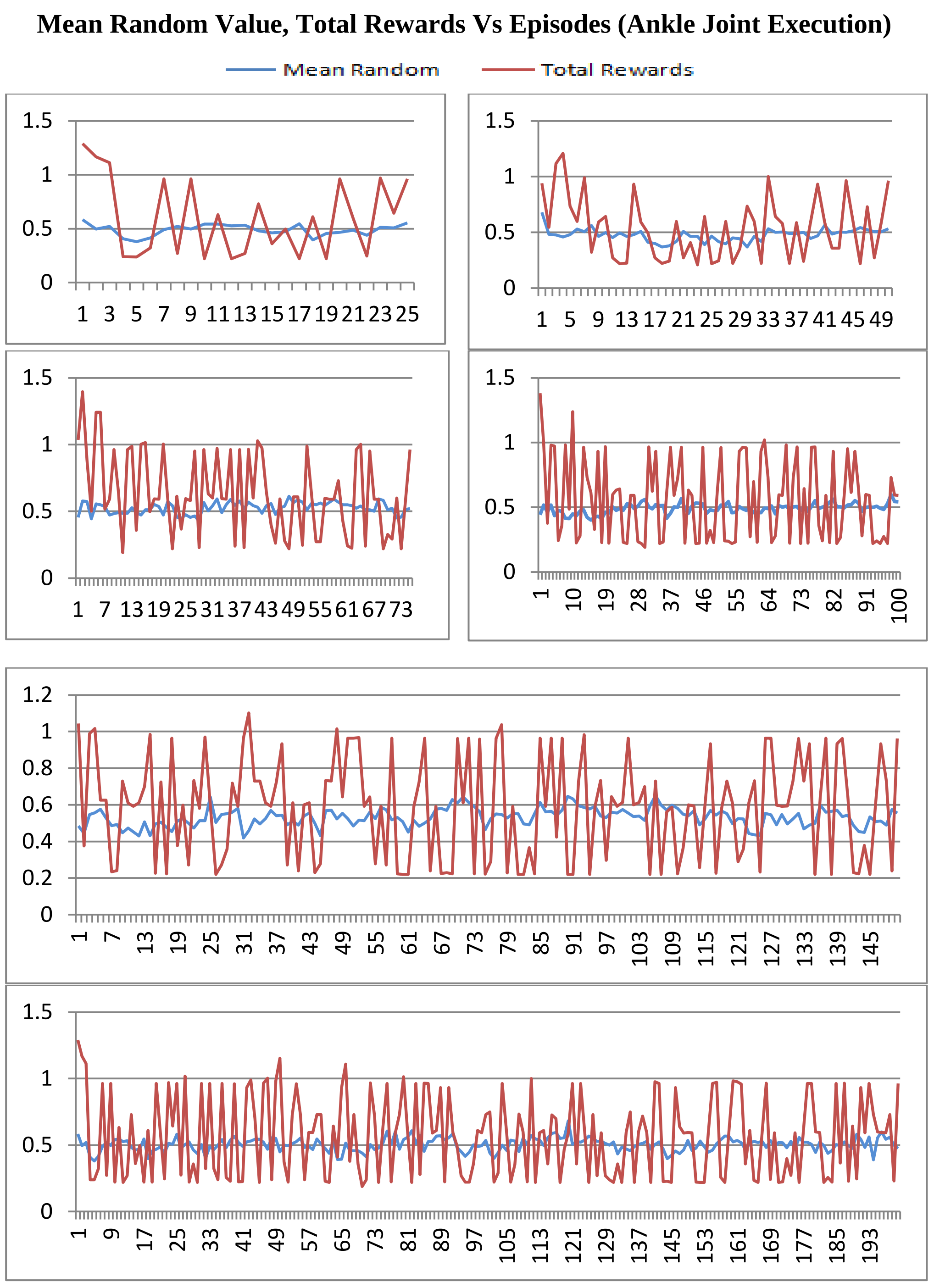


Figure 9.28 Comparison of Executing Phase of Reward Generation of Ankle Joint

9.5 Comparison of Combined Episodes for Hip, Knee and Ankle Joints in Learning and Execution Phase

The total episodes or number of iterations for the hip, knee, and ankle joint in the learning phase is evaluated by adding up all the individual numbers of iterations required to achieve the goal for each joint. Similarly, the total episodes for the execution phase are evaluated by combining the number of iterations of the hip, knee, and ankle. These values are then compared for each of 25, 50, 75, 100, 150, and 200 strides.

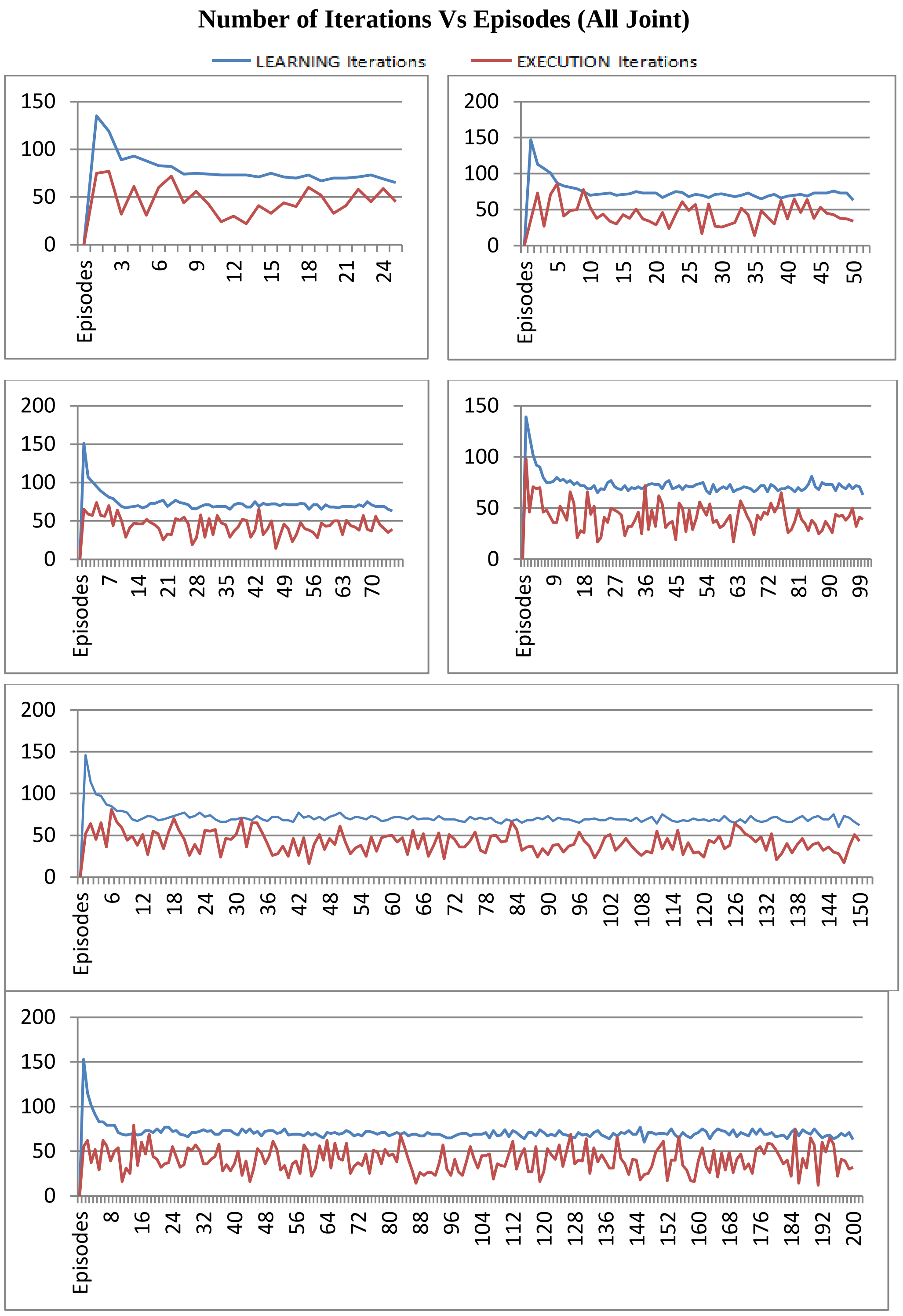
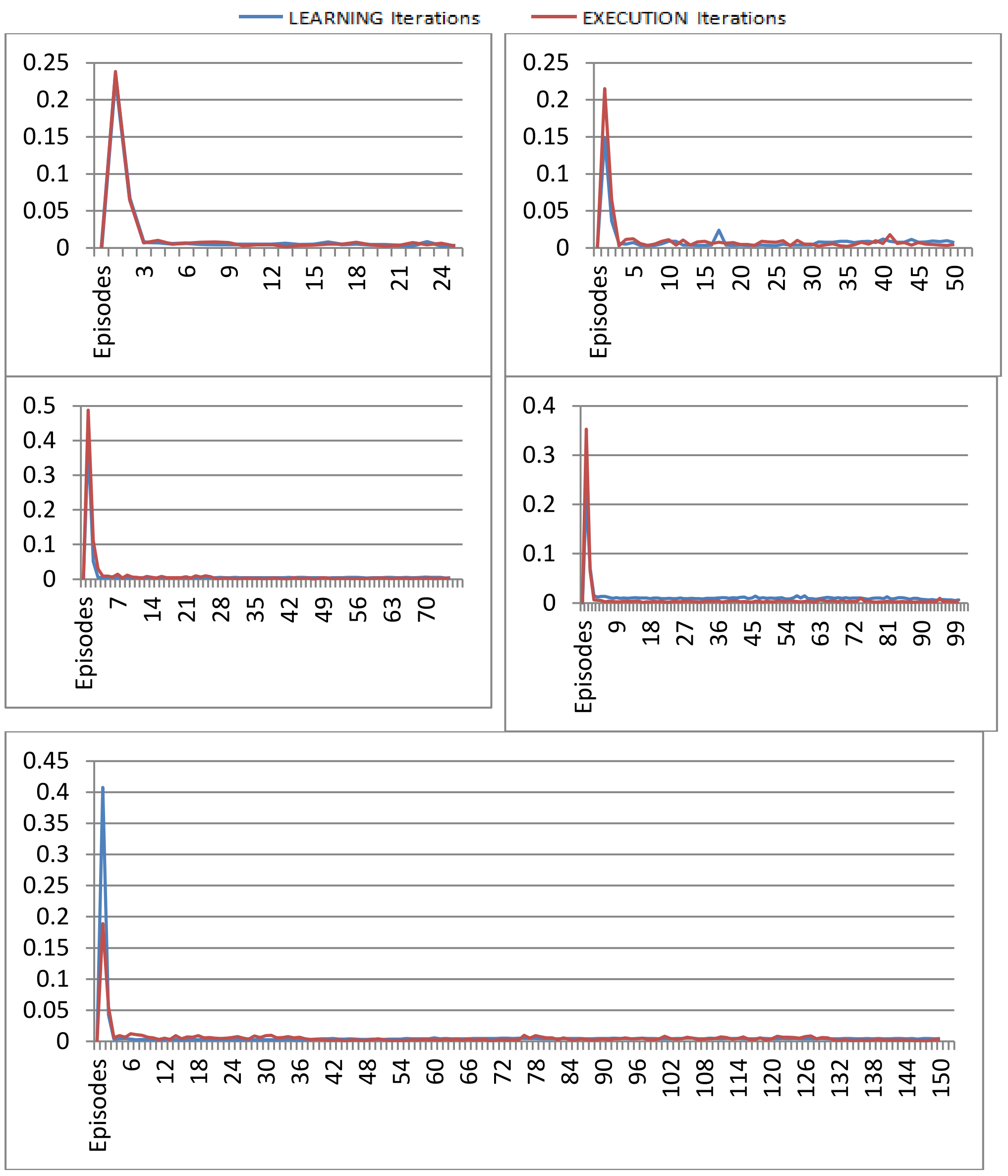


Figure 9.29 Comparison of Learning and Executing Phase of all Joint Data

9.6 Comparison of Total Time for Hip, Knee, and Ankle Joints in Learning and Execution Phase

The total time required for the learning of the hip, knee, and ankle joint in the learning phase is evaluated by adding up all the individual time required to achieve the goal for each joint. Similarly, the total time required for the execution phase is evaluated by combining the total time required for the hip, knee, and ankle. These values are then compared for each of 25, 50, 75, 100, 150, and 200 strides

Total Time Vs Episodes (All Joint)



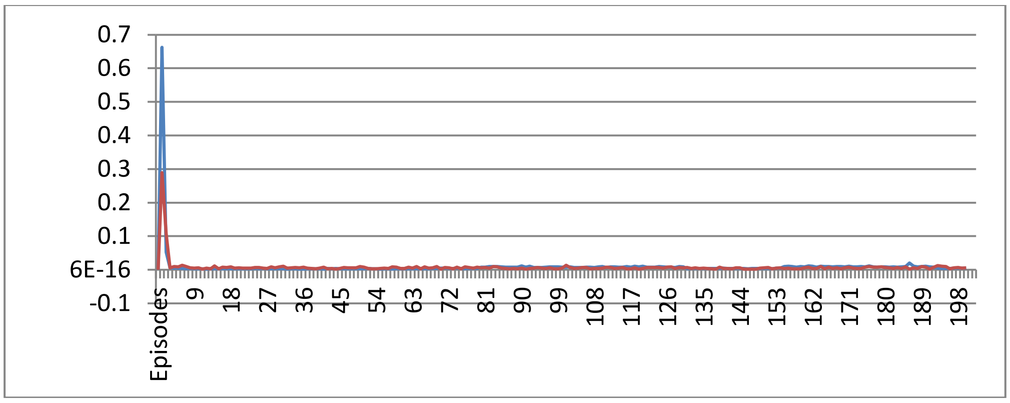


Figure 9.30 Comparison of Total Time for Learning and Executing Phase of all Joint Data

9.7 State of Art Algorithm: Computer Vision

The present work can be to some extend be compared with a computer vision state-of-the-art algorithm.

From a broader point of view, the main steps for computer vision are object classification, object identification, and object tracking.

Looking into the broader aspect, the present work has also performed these tasks along with some other tasks which makes them different from state of art algorithm.

The Bipedal first performs self-localization means finding its position in the world frame, then Bipedal identifies the object (soccer ball) using object feature detection SURF algorithm, then localizes the soccer ball concerning world frame, then calculated the distance between itself and the soccer ball. The above stated four steps are the same as performed in computer vision with a slight difference. For object identification, deep leaning or deep Q learning (DQN) algorithms are not used to reduce computation overheads and wanted to keep a clear difference between reinforcement and deep learning.

The next step of the proposed work was the reinforced learning of bipedal to walk stably and efficiently with minimal jerks and losses. For this, the forgetting mechanism was incorporated in the Q-learning algorithm. Some major changes were made in the traditional Q-leaning algorithm so that the results are as expected in the dynamic environment. The changes to list down are - incorporation of forgetting mechanism so that bipedal does not use out of date knowledge in the dynamic and uncertain environment, the rewards were not fixed/ constant but were calculated on the fly, discount factor was also changing in each go exponentially and for every 30ms the distance between the soccer ball and the bipedal is calculated. Some offline fixations are done in the algorithm and some online corrections are made to have a stable gait. In the online mode, compensators are considered along with foot adjustment for contact forces and ankle, knee, and hip joint angle calculation. After this, the bipedal has a stable gait.

Computer vision uses deep learning or DQN for these works which are supervised training algorithms.

The bipedal walk towards the soccer ball taking into consideration all dynamic and uncertain environmental conditions and stops once it reached the localized soccer ball position. At present only one bipedal is in the environment due to the constraint handling of concurrency control of bipedal.

Bipedal to some extent is using computer vision for object detection and localization as it reached the object but the method used in movement and reaching soccer ball involves a reinforcement mechanism.

CHAPTER 10 CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

The reinforcement controller for the bipedal trains the hip joint first then the knee joint is trained then the training of the ankle joint is carried out. This training helps the bipedal to move from its current state to the next stable state. After training for a stable standing position then bipedal is ready for movement searches/ looks for the object to identify in its dynamic environment. If the object is present then the bipedal calculates the distance in terms of cartesian coordinates and then walks according to the trained trajectory from the previous step, and then walks near to the identified object i.e. soccer ball in this case to kick it.

The previous knowledge is not used to train the joints of bipedal as the knowledge is outdated as the environment is dynamic and uncertain. The training of the bipedal depends on the dynamics of the system. Bipedal is trained and knowledge gained is used for exploration or exploitation steps which depends on the random value which is incorporated in the algorithm then after that, the forgetting mechanism is also implemented by setting the value of variable large so that it forgets the old information and gains new states depending on dynamics of the current system. The reward function is also not predefined or has a constant value but it is calculated on the fly by evaluating the distance between the goal and the current state and doing some algebraic and exponential operations. The bipedal is trained not to stick in any position for more than specific iterations of the training algorithm i.e. should be off the stuck position in few seconds and also trained not to change its value drastically so that it would harm the hardware of the bipedal system like servomotor, sensors, etc by passing too high or too low values to the individual parts and should not stop abruptly that is standing still or fall.

The training duration of the bipedal is not fixed it varies depending on the path followed which in turn relies on random optimized actions taken by bipedal. Bipedal was preliminary trained for 25, 50, 75, 100, 150, and 200 episodes on each joint, and the values of each iteration were stored in excel files. The individual files are created for each intermediate data and the final optimal policy reached after training is being completed by the bipedal is also recorded. Intermediate data for each episode is stored in individual sheets of an excel file so that it can be passed to the lookup table to train the bipedal joint movement individually in a hierarchical manner or maybe in a parallel manner.

Bipedal used a feature-based object identification algorithm which reduces the state space to store and hence increases the speed of object identification. In this algorithm the strongest specific amount of points are detected like 200, 400, etc then the matching is done using an affine transformation, and the updated SURF algorithm in which the image is used as an integral image and the boxlets are used to divide the image. The octave used in the pyramids helps in the filter of the increasing size usually by a factor of 2. The sign of Laplacian is used for interest point detections. Laplacian Sign helps in differentiating shiny blobs on black backgrounds and vice versa. Faster matching takes place when the same types of blobs are compared i.e. with the same type of background.

After the bipedal has learned the optimal actions and policies to take after being trained for 200 episodes for a specific dynamic environment. If bipedal runs in the same environment then it does not need to train itself again but uses the previously trained data and so in the execution phase, the iterations have been reduced from 21 of the learning phase to 2 iterations of the execution phase.

The rewards generated in learning phase varies a lot as it is randomly generated depending on the present state and target/ goal state of bipedal. But in the learning phase, bipedal is near about its goal state but that state cannot be the goal state all the time and so requires to execute some iterations but the values are near about the same as seen by the straight line as compared to the high mounted and valleyed lines of the learning phase.

This reveals that the bipedal has learned in an uncertain environment and is using that learned knowledge in the same environment. As the scenario of the environment changes the bipedal learn again, then walks to the object identified.

The bipedal identifies objects then follows the trajectory to reach the soccer ball for kicking it.

The object localization step is being performed in current work and is observed at regular interval of time to avoid a collision as soon as bipedal reaches the ball it stops to adjust the ankle and other angles to kick the ball. The main challenge in the proposed work was not object-localization but was the reinforced learning of bipedal to walk. The walk has to be stable so that the bipedal does not harm itself and the trajectory of the walk should be smooth with minimal jerks so that joint servo motors don't get damaged.
The lower body parts play a major role in the gait of the bipedal but the upper body part also has a role to play while the gait of bipedal is considered. The lower body part was considered to take into account the sub-objective of hierarchical training of all the joints (Hip, Knee, and Ankle) along with the sole placement. The complexity of the algorithm in designing the parameters was more as the reinforcement Q-Learning algorithm along with forgetting mechanism incorporation was considered. Only the upper body joints which affect the ZMP of the bipedal when running the proposed algorithm were taken into account.

10.2 Future Scope

Every work has a further step to go and each thing made has some limitations which can be improved. The bipedal has a limitation of the degree of freedom not considering the upper part of the body which also plays an important role in the gait of the bipedal.

Hands and shoulders joint and their movements are not considered. The bipedal has the limitation of the angle to move on each joint.

The designed and developed finite-state framework considers the bipedal robot as a finite state model. The bipedal has to perform self-localization as well as to object (soccer ball in this case) localization to reach the object. Bipedal on reaching the soccer ball can kick it in any direction depending on the dynamic environment. When the bipedal is playing a soccer match has to perform - self-localization, soccer ball localization, team member localization, goal post localization after these steps it would walk in the direction of the ball and try to reach it without colliding with any other player on the ground.

The autonomous bipedal system has to adapt to situations it has not previously encountered. Therefore, it needs to infer properties of its surroundings using sensors, learn from experience, and be robust to disturbances. There can be an infinite number of positions of the players on the ground and the goal post will always have a goalkeeper to reach the position in minimal time and efficiently is purpose.

The object identification is also with the limitation of identifying only the soccer ball that too in a straight line in front, other objects are not considered and in other angle directions.

The execution phase is sometimes executing 18 iterations which require a considerable amount of time and show variations in the graph of rewards generation. This can be minimized more as in some cases it reached 2 iterations but not always.

REFERENCES JOURNAL

Agarwal, S., Hyder, S., Zaidi, H., & Agarwal, S. K. (2015). Correlation of body height by foot length and knee height measurements in population of north india. *Inter*, *3*(3), 1225–1229. https://doi.org/10.16965/ijar.2015.197

Akachi, K., Kaneko, K., Kanehira, N., Ota, S., Miyamori, G., Hirata, M., Kajita, S., & Kanehiro, F. (2005). Development of humanoid robot HRP-3P. *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, *2005*, 50–55. https://doi.org/10.1109/ICHR.2005.1573544

A. Arunmozhi and J. Park(2018), Comparison of HOG, LBP and Haar-Like Features for On-Road Vehicle Detection, *2018 IEEE International Conference on Electro/Information Technology (EIT),* Rochester, MI, 2018, pp. 0362-0367, doi: 10.1109/EIT.2018.8500159.

Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R. R., Bluethmann, W., Diftler, M., Magruder, D., Rehnmark, F., & Johnson, N. (1973). *Robonaut : NASA ' s Space Humanoid*.

Asfour, T., Azad, P., Vahrenkamp, N., Regenstein, K., Bierbaum, A., Welke, K., Schröder, J., & Dillmann, R. (2008). Toward humanoid manipulation in human-centred environments. *Robotics and Autonomous Systems*, *56*(1), 54–65. https://doi.org/10.1016/j.robot.2007.09.013

Asfour, T., Regenstein, K., Azad, P., Schröder, J., Bierbaum, A., Vahrenkamp, N., & Dillmann, R. (2006). ARMAR-III: An integrated humanoid platform for sensory-motor control. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, 169–175. https://doi.org/10.1109/ICHR.2006.321380

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. *34th International Conference on Machine Learning, ICML 2017*.

Bharadwaj, D., Prateek, M., & Sharma, R. (2019). Development of reinforcement control algorithm of lower body of autonomous humanoid robot. *International Journal of Recent Technology and Engineering*, *8*(1), 915–919.

Borst, C., Ott, C., Wimbck, T., Brunner, B., Zacharias, F., B??uml, B., Hillenbrand, U., Haddadin, S., Albu-Sch??ffer, A., & Hirzinger, G. (2007). A humanoid upper body system for two-handed manipulation. *Proceedings -*

IEEE International Conference on Robotics and Automation, *April*, 2766–2767. https://doi.org/10.1109/ROBOT.2007.363886

Brooks, R. A. (1987). A hardware retargetable distributed layered architecture for mobile robotcontrol. *CH2413-3/87/3000/0106~3~.03 0 1987 IEEE*.

Burghart, C., Mikut, R., Stiefelhagen, R., Asfour, T., Holzapfel, H., Steinhaus, P., & Dillmann, R. (2005). A Cognitive Architecture for a Humanoid Robot : A First Approach. *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 357–362. https://doi.org/10.1109/ICHR.2005.1573593

Caldwel, D. G., & Bowler, C. J. (1997). Investigation of Bipedal Robot Locomotion using Pneumatic Muscle Actuators. *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*.

Cenciarini, M., & Dollar, A. M. (2011). Biomechanical considerations in the design of lower limb exoskeletons. *IEEE International Conference on Rehabilitation Robotics*, 1–6. https://doi.org/10.1109/ICORR.2011.5975366

Christen, S., & Stevˇ, S. (2019). Demonstration-Guided Deep Reinforcement Learning of Control Policies for Dexterous Human-Robot Interaction. *AIT Lab, Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland*, *i*, 2161–2167.

Copyright, F. R., Company, F. S., & Only, F. E. (2007). Integrated Motion Control. *Proceedings of Lthe 2004 IEEE Lnternatlonal Conference on Robotics & Automation*, *C*, 2005–2007.

Dahl, T., & Boulos, M. (2013). Robots in Health and Social Care: A Complementary Technology to Home Care and Telehealthcare? *Robotics*, *3*(1), 1–21. https://doi.org/10.3390/robotics3010001

Danel, M. (2017). Reinforcement Learning for Humanoid Robot Control. *POSTER,PRAGUE*, 1–5.

Duan, Y., Liu, Q., & Xu, X. H. (2007). Application of reinforcement learning in robot soccer. *Engineering Applications of Artificial Intelligence*. https://doi.org/10.1016/j.engappai.2007.01.003

Endo, N., Momoki, S., Zecca, M., Saito, M., Mizoguchi, Y., Itoh, K., & Takanishim, A. (2008). Development of whole-body emotion expression humanoid robot. *Proceedings - IEEE International Conference on Robotics and Automation*, 2140–2145. https://doi.org/10.1109/ROBOT.2008.4543523

Erhart, S., Sieber, D., & Hirche, S. (2013). An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation. *IEEE*

International Conference on Intelligent Robots and Systems, 315–322. https://doi.org/10.1109/IROS.2013.6696370

Feil-Seifer, D., & Matarić, M. J. (2008). B3IA: A control architecture for autonomous robot-assisted behavior intervention for children with autism spectrum disorders. *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN*, 328–333. https://doi.org/10.1109/ROMAN.2008.4600687

Frank, M., Leitner, J., Stollenga, M., Forster, A., & Schmidhuber, J. (2014). Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in Neurorobotics*, *7*(JAN), 1–15. https://doi.org/10.3389/fnbot.2013.00025

Fukaya, N., & Toyama, S. (2000). Design of the TUAT / Karlsruhe Humanoid Hand. *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, *March*, 1754–1759.

Gabel, T., & Riedmiller, M. (2012). Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2011.571443

Galindo, C., Gonzalez, J., & Fernández-Madrigal, J. A. (2006). Control architecture for human-robot integration: Application to a robotic wheelchair. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *36*(5), 1053–1067. https://doi.org/10.1109/TSMCB.2006.874131

Garofalo, G., Ott, C., & Albu-Schäffer, A. (2012). Walking control of fully actuated robots based on the bipedal SLIP model. *Proceedings - IEEE International Conference on Robotics and Automation*, 1456–1463. https://doi.org/10.1109/ICRA.2012.6225272

Ghavamzadeh, M., & Mahadevan, S. (2007). Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research*.

Gienger, M., Loffler, K., & Pfeiffer, F. (2002). *Towards the design of a biped jogging robot*. 4140–4145. https://doi.org/10.1109/robot.2001.933265

Grizzle, J. W., Hurst, J., Morris, B., Park, H., & Sreenath, K. (2009). MABEL , A New Robotic Bipedal Walker and Runner. *2009 American Control Conference*, 2030–2036. https://doi.org/10.1109/ACC.2009.5160550

Guenter, F., Hersch, M., Calinon, S., & Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, *21*(13), 1521–1544. https://doi.org/10.1163/156855307782148550

Ha, I., Tamura, Y., Asama, H., Han, J., & Hong, D. W. (2011). Development of open humanoid platform DARwIn-OP. *Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2178–2181.

Hernández-Santos, C., Rodriguez-Leal, E., Soto, R., & Gordillo, J. L. (2012). Kinematics and dynamics of a new 16 DOF humanoid biped robot with active toe joint. *International Journal of Advanced Robotic Systems*, *9*. https://doi.org/10.5772/52452

Hester, T., Quinlan, M., & Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. *Proceedings - IEEE International Conference on Robotics and Automation*, *May*, 2369–2374. https://doi.org/10.1109/ROBOT.2010.5509181

Huang, B. Q., Cao, G. Y., & Guo, M. (2005). Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. *2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005*. https://doi.org/10.1109/icmlc.2005.1526924

Huang, Y., Vanderborght, B., Van Ham, R., Wang, Q., Van Damme, M., Xie, G., & Lefeber, D. (2013). Step length and velocity control of a dynamic bipedal walking robot with adaptable compliant joints. *IEEE/ASME Transactions on Mechatronics*, *18*(2), 598–611. https://doi.org/10.1109/TMECH.2012.2213608

Iida, S. (2004). Humanoid Robot Control Based on Reinforcement Learning. *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004.*, 353–358. https://doi.org/10.1109/MHS.2004.1421274

Inada, H. (2003). Behavior Generation of Bipedal Robot Using Central Pattern Generator(CPG) (1st Report: CPG Parameters Searching Method by Genetic Algorithm). *Proceedings 01 the 2003 IEEWRSJ Lntl Conterence on Ntell Gent Robots Ana Systems Las*, *October*, 2179–2184.

Inoue, S., & Takanishi, A. (1999). Development of a Bipedal Humanoid Robot - Control Method of Whole Body Cooperative Dynamic Biped Walking -. *Proceedings of the 1999 EEE International Conference on Robotics & Automation Detroit, Michigan May 1999 -*, *May*, 368–374.

Isbell, J., Shelton, C. R., Kearns, M., Singh, S., & Stone, P. (2001). A social reinforcement learning agent. *Proceedings of the International Conference on Autonomous Agents*. https://doi.org/10.1145/375735.376334

Iwata, H., & Sugano, S. (2009). Design of human symbiotic robot TWENDY-ONE. *Proceedings - IEEE International Conference on Robotics and Automation*, 580–586. https://doi.org/10.1109/ROBOT.2009.5152702

Janssens, D., Lan, Y., Wets, G., & Chen, G. (2007). Allocating time and location information to activity-travel patterns through reinforcement learning. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2007.01.008

Kagami, S., Nishiwaki, K., Kuffner, J. J., Kuniyoshi, Y., Inaba, M., & Inoue, H. (2003). *Online 3D vision, motion planning and bipedal locomotion control coupling system of humanoid robot: H7*. *October*, 2557–2562. https://doi.org/10.1109/irds.2002.1041655

Kanda, T., Hiroshi, I., Imai, M., Ono, T., & Mase, K. (2002). A constructive approach for developing interactive humanoid robots. *Proceedings of the 2002 IEEElRSJ Intl. Conference on Intelligent Robots and Systems EPFL, Lausanne, Switzerland, October*, 1265–1270.

Kanehira, N., Kawasaki, T. U., Ohta, S., Ismumi, T., Kawada, T., Kanehiro, F., Kajita, S., & Kaneko, K. (2003). *Design and experiments of advanced leg module (HRP-2L) for humanoid robot (HRP-2) development*. *2*(October), 2455–2460. https://doi.org/10.1109/irds.2002.1041636

Kareem Jaradat, M. A., Al-Rousan, M., & Quadan, L. (2011). Reinforcement based mobile robot navigation in dynamic environment. *Robotics and Computer-Integrated Manufacturing*. https://doi.org/10.1016/j.rcim.2010.06.019

Kartoun, U., Stern, H., & Edan, Y. (2010). A human-robot collaborative reinforcement learning algorithm. *Journal of Intelligent and Robotic Systems: Theory and Applications, 60*(2), 217–239. https://doi.org/10.1007/s10846-010-9422-y

Kati, D., & Vukobratovi, M. (2006). Control Algorithm for Humanoid Walking Based on Fuzzy Reinforcement Learning Model of the Robot ' s Mechanism. *SISY 2006 • 4th Serbian-Hungarian Joint Symposium on Intelligent Systems*, 81–93.

Katic, D., & Vukobratovic, M. (n.d.). Intelligent control techniques for humanoid robots. *Robotics Laboratory, Mihailo Pupin Institute P.O.Box 15, Volgina 15, 11000 Belgrade, Yugoslavia*, 1839–1844. https://doi.org/10.23919/ecc.2003.7085233

Ken'ichiro, N. (1997). Acquisition of Visually Guided Swing Motion Based on Genetic Algorithms and Neural Networks in Two-Armed Bipedal.

Proceedings of the 1997 IEEE Lnternational Conference on Robotics and Automation, *April*, 2944–2949.

Khatib, O. (1987). A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation. *IEEE Journal on Robotics and Automation*, *3*(1), 43–53. https://doi.org/10.1109/JRA.1987.1087068

Khatib, O. (1999). Robotics and Autonomous Systems Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, *26*, 175–183.

Kim, J., Lee, Y., Kwon, S., Seo, K., Kwak, H., Lee, H., & Roh, K. (2012). Development of the Lower Limbs for a Humanoid Robot. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4000–4005. https://doi.org/10.1109/IROS.2012.6385728

Kim, J. Y., Park, I. W., Lee, J., Kim, M. S., Cho, B. K., & Oh, J. H. (2005). System design and dynamic walking of humanoid robot KHR-2. *Proceedings - IEEE International Conference on Robotics and Automation*, *2005*(April), 1431–1436. https://doi.org/10.1109/ROBOT.2005.1570316

Kim, S. C. Y., & Hutchinson, S. (2008). An Improved Hierarchical Motion Planner for Humanoid Robots. *2008 8th IEEE-RAS International Conference on Humanoid Robots December 1 -- 3, 2008 / Daejeon, Korea*, 654–661.

Kim, S. K., Kirchner, E. A., Stefes, A., & Kirchner, F. (2017). Intrinsic interactive reinforcement learning-Using error-related potentials for real world human-robot interaction. *Scientific Reports*, *7*(1), 1–16. https://doi.org/10.1038/s41598-017-17682-7

Kober, J., Oztop, E., & Peters, J. (2011). Reinforcement learning to adjust robot movements to new situations. *IJCAI International Joint Conference on Artificial Intelligence*, 2650–2655. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-441

Kober, J., & Peters, J. (2006). PolicySearchforMotorPrimitivesinRobotics. *Max Planck Institute for Biological Cybernetics Spemannstr.38 72076 Tübingen, Germany*, *2*(1), 87–99. https://doi.org/10.2217/1745509X.2.1.87

Komatsu, T. (2005). Dynamic Walking and Running of a Bipedal Robot Using Hybrid Central Pattern Generator Method. *IEEE International Conference Mechatronics and Automation, 2005*, *2*(July), 987-992 Vol. 2. https://doi.org/10.1109/ICMA.2005.1626686

Kondo, Y., Takemura, K., Takamatsu, J., & Ogasawara, T. (2013). A Gesture-Centric Android System for Multi-Party Human-Robot Interaction. *Journal of*

Human-Robot Interaction, *2*(1), 133–151. https://doi.org/10.5898/jhri.2.1.kondo

Kuffner, J. J. (2001). Footstep Planning Among Obstacles for Biped Robots. *Proeocdings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and System Maui, Hawaii, USA*, 500–505.

Kuroki, Y., Kato, K., Nagasaka, K., Miyamoto, A., Ueno, K., & Yamaguchi, J. (2004). *Motion creating system for a small biped entertainment robot*. *October*, 3809-3814 Vol.4. https://doi.org/10.1109/robot.2004.1308862

Kuroki, Yoshihiro, Fujita, M., Ishida, T., Nagasaka, K., & I, J. Y. (2003). A Small Biped Entertainment Robot Exploring Attractive Applications. *Proceedings of the 2003 IEEE Lnternatiooal Conference on Robotics & Automation*, 471–476.

Lapeyre, M., N'Guyen, S., Le Falher, A., & Oudeyer, P. Y. (2015). Rapid morphological exploration with the Poppy humanoid platform. *IEEE-RAS International Conference on Humanoid Robots*, *2015-Febru*, 959–966. https://doi.org/10.1109/HUMANOIDS.2014.7041479

Lee, J. H., & Labadie, J. W. (2007). Stochastic optimization of multireservoir systems via reinforcement learning. *Water Resources Research*. https://doi.org/10.1029/2006WR005627

Lim, S. C., & Yeap, G. H. (2012). The locomotion of bipedal walking robot with six degree of freedom. *Procedia Engineering*, *41*(Iris), 8–14. https://doi.org/10.1016/j.proeng.2012.07.136

Ling, K., & Shalaby, A. S. (2005). A reinforcement learning approach to streetcar bunching control. *Journal of Intelligent Transportation Systems*. https://doi.org/10.1080/15472450590934615

Liu, H., Iberall, T., & Bekey, G. A. (1989). Neural Network Architecture for Robot Hand Control. *IEEE Control Systems Magazine*, *5*(April), 183–190.

Lober, R., Padois, V., Sigaud, O., Lober, R., Padois, V., Sigaud, O., Reinforcement, E., Lober, R., Padois, V., & Sigaud, O. (2016). Efficient Reinforcement Learning for Humanoid Whole-Body Control. *IEEE-RAS International Conference on Humanoid Robots, Nov 2016, Cancun, Mexico. Hal-01377831 HAL*.

Lohmeier, S., Buschmann, T., & Ulbrich, H. (2009). Humanoid robot LOLA. *Proceedings - IEEE International Conference on Robotics and Automation*, 775–780. https://doi.org/10.1109/ROBOT.2009.5152578

Lowrey, K., Kolev, S., Dao, J., Rajeswaran, A., & Todorov, E. (2018). Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR 2018*, 35–42. https://doi.org/10.1109/SIMPAR.2018.8376268

Lundberg, I., Björkman, M., & Ögren, P. (2015). Intrinsic camera and hand-eye calibration for a robot vision system using a point marker. *IEEE-RAS International Conference on Humanoid Robots*. https://doi.org/10.1109/HUMANOIDS.2014.7041338

Ly, D. N., Regenstein, K., & Asfour, T. (2004). A Modular and Distributed Embedded Control Architecture for Humanoid Robots. *Proceedings 01 2004 IEEWRSJ Lnternstional Conference on IntelligentRobots and Systems Sendai, Japan*, 2775–2780.

Mahadevan, S. (1996). Average reward reinforcement learning: foundations, algorithms, and empirical results. *Machine Learning*. https://doi.org/10.1007/BF00114727

Maniatopoulos, S., Schillinger, P., Pong, V., Conner, D. C., & Kress-gazit, H. (2016). Reactive High-level Behavior Synthesis for an Atlas Humanoid Robot. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 4192–4199. https://doi.org/10.1109/ICRA.2016.7487613

Mahadevan, S.(1996) Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Mach Learn* **22,** 159–195. https://doi.org/10.1007/BF00114727

Mansard, N., Stasse, O., Evrard, P., & Kheddar, A. (2009). A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks. *Advanced Robotics, 2009. ICAR 2009. International Conference On, 8,* 1–6.

Maravall, D., De Lope, J., & Domínguez, R. (2013). Coordination of communication in robot teams by reinforcement learning. *Robotics and Autonomous Systems*. https://doi.org/10.1016/j.robot.2012.09.016

Martinez-Cantin, R., De Freitas, N., Brochu, E., Castellanos, J., & Doucet, A. (2009). A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*. https://doi.org/10.1007/s10514-009-9130-2

Matarić, M. J. (1997). Reinforcement Learning in the Multi-Robot Domain. *Autonomous Robots*. https://doi.org/10.1023/A:1008819414322

Matsubara, T., Shinohara, D., & Kidode, M. (2013). Reinforcement learning of a motor skill for wearing a T-shirt using topology coordinates. *Advanced Robotics*. https://doi.org/10.1080/01691864.2013.777012

Maurtua, I., Ibarguren, A., Kildal, J., Susperregi, L., & Sierra, B. (2017). Human-robot collaboration in industrial applications: Safety, interaction and trust. *International Journal of Advanced Robotic Systems*, *14*(4), 1–10. https://doi.org/10.1177/1729881417716010

Michel, P., Chestnutt, J., Kuffner, J., & Kanade, T. (2005). Vision-guided humanoid footstep planning for dynamic environments. *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, *2005*, 13–18. https://doi.org/10.1109/ICHR.2005.1573538

Mohamed, Z., & Capi, G. (2012). *Development of a New Mobile Humanoid Robot for Assisting Elderly People*. *41*(Iris), 345–351. https://doi.org/10.1016/j.proeng.2012.07.183

Motor, H. (2007). Asimo. *Public Relations Division*, *September*.

Nanduri, V., & Das, T. K. (2009). A reinforcement learning algorithm for obtaining the Nash equilibrium of multi-player matrix games. *IIE Transactions (Institute of Industrial Engineers)*. https://doi.org/10.1080/07408170802369417

Naumann, M., Wegener, K., Schraft, R. D., & Ipa, F. (2007). Control Architecture for Robot Cells to Enable Plug ' n ' Produce. *IEEE International Conference on Robotics and Automation*, *April*, 10–14.

Nelson, G., Saunders, A., Neville, N., Swilling, B., Bondaryk, J., Billings, D., Lee, C., Playter, R., & Raibert, M. (2012). PETMAN: A Humanoid Robot for Testing Chemical Protective Clothing. *Journal of the Robotics Society of Japan*, *30*(4), 372–377. https://doi.org/10.7210/jrsj.30.372

Ng, A. (2012). 12. Reinforcement Learning and Control. *Machine Learning*.

Niiyama, R., Nishikawa, S., & Kuniyoshi, Y. (2010). Athlete Robot with Applied Human Muscle Activation Patterns for Bipedal Running. *2010 10th IEEE-RAS International Conference on Humanoid Robots*, 498–503. https://doi.org/10.1109/ICHR.2010.5686316

Nishino, D., & Takanishi, A. (1998). Development of a Bipedal Humanoid Robot Having Antagonistic Driven Joints and Three DOF Trunk. *Roceedings of the 1998 IEEE/RSJ Htl. Conference on Intelligent Robots and Systems Victoria, B.C., Canada October 1998*, *October*, 96–101.

Ogura, Y., Aikawa, H., Shimomura, K., Kondo, H., Morishima, A., Lim, H. O., & Takanishi, A. (2006). Development of a new humanoid robot WABIAN-2. *Proceedings - IEEE International Conference on Robotics and Automation*, *2006*(May), 76–81. https://doi.org/10.1109/ROBOT.2006.1641164

Oh, J. H., Hanson, D., Kim, W. S., Han, I. Y., Kim, J. Y., & Park, I. W. (2006). Design of android type humanoid robot Albert HUBO. *IEEE International Conference on Intelligent Robots and Systems*, 1428–1433. https://doi.org/10.1109/IROS.2006.281935

Okada, K., Kojima, M., Sagawa, Y., Ichino, T., Sato, K., & Inaba, M. (2006). Vision based behavior verification system of humanoid robot for daily environment tasks. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, *00*, 7–12. https://doi.org/10.1109/ICHR.2006.321356

Ott, C., Eiberger, O., Friedl, W., Bäuml, B., Hillenbrand, U., Borst, C., Albu-Schäffer, A., Brunner, B., Hirschmüller, H., Kielhöfer, S., Konietschke, R., Suppa, M., Wimböck, T., Zacharias, F., & Hirzinger, G. (2006). A humanoid two-arm system for dexterous manipulation. *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, 276–283. https://doi.org/10.1109/ICHR.2006.321397

Palmer, V. (2007). Scaling reinforcement learning to the unconstrained multi-agent domain. *ProQuest Dissertations and Theses*.

Pandey, A. K., Gelin, R., Alami, R., Viry, R., Buendia, A., Meertens, R., Chetouani, M., Devillers, L., Tahon, M., Filliat, D., Grenier, Y., Maazaoui, M., Kheddar, A., Lerasle, F., & Duval, L. F. (2014). Romeo2 project: Humanoid robot assistant and companion for everyday life: I. situation assessment for social intelligence. *CEUR Workshop Proceedings*, *1315*(November), 140–147.

Park, I. W., Kim, J. Y., Lee, J., & Oh, J. H. (2005). Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot - 3: HUBO). *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, *2005*, 321–326. https://doi.org/10.1109/ICHR.2005.1573587

Park, J. (2007). General ZMP Preview Control for Bipedal Walking. *EEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007*, *April*, 10–14.

Pateromichelakis, N., Mazel, A., Hache, M. A., Koumpogiannis, T., Gelin, R., Maisonnier, B., & Berthoz, A. (2014). Head-eyes system and gaze analysis of the humanoid robot Romeo. *IEEE International Conference on Intelligent Robots and Systems, Iros*, 1374–1379. https://doi.org/10.1109/IROS.2014.6942736

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement Learning for Humanoid Robotics. *Third IEEE-RAS International Conference on Humanoid Robots, Karlsruhe, Germany*.

Pontrandolfo, P., Gosavi, A., Okogbaa, O. G., & Das, T. K. (2002). Global supply chain management: A reinforcement learning approach. *International Journal of Production Research*. https://doi.org/10.1080/00207540110118640

Posadas, J. L., Poza, J. L., Simó, J. E., Benet, G., & Blanes, F. (2008). Agent-based distributed architecture for mobile robot control. *Engineering Applications of Artificial Intelligence*, *21*(6), 805–823. https://doi.org/10.1016/j.engappai.2007.07.008

Quintía, P., Iglesias, R., Rodrguez, M. A., & Regueiro, C. V. (2010). Simultaneous learning of perception and action in mobile robots. *Robotics and Autonomous Systems*. https://doi.org/10.1016/j.robot.2010.08.009

Reil, T., & Husbands, P. (2002). Evolution of Central Pattern Generators for Bipedal Walking in a Real-Time Physics Environment. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6, NO. 2, APRIL 2002*, *6*(2), 159–168.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *31st International Conference on Machine Learning, ICML 2014*.

Riedmiller, M., Gabel, T., Hafner, R., & Lange, S. (2009). Reinforcement learning for robot soccer. *Autonomous Robots*, *27*(1), 55–73. https://doi.org/10.1007/s10514-009-9120-4

Robinson, H., MacDonald, B., & Broadbent, E. (2014). The Role of Healthcare Robots for Older People at Home: A Review. *International Journal of Social Robotics*, *6*(4), 575–591. https://doi.org/10.1007/s12369-014-0242-2

Rohmer, E., Singh, S. P. N., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. *IEEE International Conference on Intelligent Robots and Systems*, 1321–1326. https://doi.org/10.1109/IROS.2013.6696520

Rosenblatt, J. K., & Payton, D. W. (n.d.). A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control. *Hughes Artificial*

Intelligence Center Hughes Research Laboratories 301 1 Malibu Canyon Road Malibu, California 90265.

Rostami, M., & Bessonnet, G. (1998). Impactless sagittal gait of a biped robot during the single support phase. *Proceedings of the 1998 IEEE International Conference on Robotics L? Automation*, *May*, 1385–1391.

Routray, S., Ray, A. K., & Mishra, C. (2017). Analysis of various image feature extraction methods against noisy image: SIFT, SURF and HOG. *Proceedings of the 2017 2nd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2017*. https://doi.org/10.1109/ICECCT.2017.8117846

Sandon, F., Bush, R. R., & Mosteller, F. (1956). Stochastic Models for Learning. *The Mathematical Gazette*. https://doi.org/10.2307/3609656

Sawasaki, N., Nakajima, T., Shiraishi, A., Nakamura, S., Wakabayashi, K., & Sugawara, Y. (2004). *Application of humanoid robots to building and home management services*. 2992–2997. https://doi.org/10.1109/robot.2003.1242050

Schwartz, A. (1993). A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *Machine Learning Proceedings 1993*. https://doi.org/10.1016/b978-1-55860-307-3.50045-9

Shamsuddin, S., Ismail, L. I., Yussof, H., Zahari, N. I., & Bahari, S. (2011). Humanoid Robot NAO : Review of Control and Motion Exploration. *2011 IEEE International Conference on Control System, Computing and Engineering*, 511–516. https://doi.org/10.1109/ICCSCE.2011.6190579

Shamsuddin, S., Yussof, H., Robots, H., & Hurobs, B. (2012). Initial Response of Autistic Children in Human-Robot Interaction Therapy with Humanoid Robot NAO. *2012 IEEE 8th International Colloquium on Signal Processing and Its Applications*, 188–193. https://doi.org/10.1109/CSPA.2012.6194716

Sharma, R., Prateek, M., & K. Sinha, A. (2013). Use of Reinforcement Learning as a Challenge: A Review. *International Journal of Computer Applications*, *69*(22), 28–34. https://doi.org/10.5120/12105-8332

Sharma, R., Singh, I., Bharadwaj, D., & Prateek, M. (2019). Incorporating forgetting mechanism in q-learning algorithm for locomotion of bipedal walking robot. *International Journal of Innovative Technology and Exploring Engineering*, *8*(7), 1782–1787.

Sharma, R., Singh, I., Prateek, M., & Pasricha, A. (2019). Implementation of feature based object identification in Bipedal walking robot. *International Journal of Engineering and Advanced Technology*, *8*(5), 110–113.

Sharma, R., Singh, I., Prateek, M., & Pasricha, A. (2020). Comparative study of learning and execution of bipedal by using forgetting mechanism in reinforcement learning algorithm. *Journal Europeen Des Systemes Automatises*, *53*(3), 335–343. https://doi.org/10.18280/jesa.530304

Shokri, M. (2011). Knowledge of opposite actions for reinforcement learning. *Applied Soft Computing Journal*. https://doi.org/10.1016/j.asoc.2011.01.045

Silva, I. J., Perico, D. H., Costa, A. H., & Bianchi, R. A. (2017). Using Reinforcement Learning To Optimize Gait Generation. *XIII Simp´osio Brasileiro de Automac¸a˜o Inteligente*, 288–294.

Silva, M., Barbosa, R., & Castro, T. (2015). Multi-legged walking robot modelling in MATLAB/simmechanics TM and its simulation. *Proceedings - 8th EUROSIM Congress on Modelling and Simulation, EUROSIM 2013*, 226–231. https://doi.org/10.1109/EUROSIM.2013.50

Simmons, R., & Apfelbaum, D. (1998). Task Description Language for Robot Control. *Proceedings of the 1998 IEEEYRSJ Intl. Conference on Intelligent Robots and Systems, October*, 1931–1937.

Singh, S., Barto, A. G., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems*. https://doi.org/10.21236/ada440280

Sko, D. U. (2008). Reinforcement learning control algorithm for humanoid robot walking. *International journal of Information of Informatics and systems sciences*, *4*(2), 256–267.

Song, A., Song, G., Constantinescu, D., Wang, L., & Song, Q. (2015). Sensors for Robotics 2015. *Journal of Sensors*, *2015*, 1–2. https://doi.org/10.1155/2015/412626

Sternberg, R. J., & Kaufman, J. C. (2016). Intelligence. In *The Curated Reference Collection in Neuroscience and Biobehavioral Psychology*. https://doi.org/10.1016/B978-0-12-809324-5.03075-3

Stone, P., & Sutton, R. (2001). Scaling reinforcement learning toward RoboCup soccer. *Icml*.

Sutton, R. S. (1990). Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *Machine*

Learning Proceedings 1990. https://doi.org/10.1016/b978-1-55860-141-3.50030-4

Sutton, R. S., & Barto, A. G. (2012). Reinforcement learning: An Introduction Second edition. *Learning*.

Tamei, T., & Shibata, T. (2011). Fast reinforcement learning for three-dimensional kinetic human-robot cooperation with an EMG-to-activation model. *Advanced Robotics*. https://doi.org/10.1163/016918611X558252

Tanaka, Y., Lee, H., Wallace, D., Jun, Y., Oh, P., & Inaba, M. (2017). Toward deep space humanoid robotics inspired by the NASA Space Robotics Challenge. *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017*, 14–19. https://doi.org/10.1109/URAI.2017.7992877

Tehrani, A. M., & Kamel, M. S. (2005). Behavior arbitration using a fuzzy reinforcement learning approach. *Intelligent Automation and Soft Computing*. https://doi.org/10.1080/10798587.2005.10642903

Tellez, R., Ferro, F., Garcia, S., Golnez, E., Jorge, E., Mora, D., Pinyo, D., Oliver, J., Torres, O., Velazquez, J., & Faconti, D. (2008). Reem-B: an autonomous lightweight human-size humanoid robot. *2008 8th IEEE-RAS International Conference on Humanoid Robots December 1 -- 3, 2008/ Daejeon, Korea WP1-25*, 462–468.

Thuilot, B., Goswami, A., & Espiau, B. (2002). Bifurcation and chaos in a simple passive bipedal gait. *Proceedings of the 1997 IEEE International Conference on Robotics and Automation Albuquerque, New Mexico - April 1997 Bifurcation*, *April*, 792–798. https://doi.org/10.1109/robot.1997.620131

Tikhanoff, V., Fitzpatrick, P., Nori, F., Natale, L., Metta, G., & Cangelosi, A. (n.d.). *The iCub Humanoid Robot Simulator*. *1*.

Tlalolini, D., Chevallereau, C., & Aoustin, Y. (2011). Human-Like Walking : Optimal Motion of a Bipedal Robot With Toe-Rotation Motion. *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, *16*(2), 310–320.

Tsagarakis, N. G., Li, Z., Saglia, J., & Caldwell, D. G. (2011). The design of the lower body of the compliant humanoid robot “cCub.” *Proceedings - IEEE International Conference on Robotics and Automation*, 2035–2040. https://doi.org/10.1109/ICRA.2011.5980130

Tesauro, G. J., (1995) Temporal difference learning and TDGammon.Commun. ACM 38, 58–68 (1995).

Ueda, J., Negi, R., & Yoshikawa, T. (2004). Acquisition of a page turning skill for a multifingered hand using reinforcement learning. *Advanced Robotics*. https://doi.org/10.1163/156855304322753326

Velentzas, G., Tsitsimis, T., Rañó, I., Tzafestas, C., & Khamassi, M. (2018). Adaptive reinforcement learning with active state-specific exploration for engagement maximization during simulated child-robot interaction. *Paladyn*, *9*(1), 235–253. https://doi.org/10.1515/pjbr-2018-0016

Wang, Y., & Butner, S. E. (1987). A New Architecture for Robot Control. *CH2413-3/87/0000/0664$01.00 Q 1987 IEEE 664*, *08421415*, 664–670.

Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*. https://doi.org/10.1016/j.engappai.2004.08.018

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*. https://doi.org/10.1007/bf00992698

Wawrzyński, P. (2012). Autonomous reinforcement learning with experience replay for humanoid gait optimization. *Procedia Computer Science*, *13*, 205–211. https://doi.org/10.1016/j.procs.2012.09.130

Weiß, G. (1995). Distributed reinforcement learning. *Robotics and Autonomous Systems*. https://doi.org/10.1016/0921-8890(95)00018-B

Wyeth, G., Kee, D., Wagstaff, M., Brewer, N., & Art, P. (2001). Design of an Autonomous Humanoid Robot. *Proc.2001 Australian Conference on Robotics and Automation,Sydney, November*, 14–15.

Yamasaki, F., Matsui, T., Miyashita, T., & Kitano, H. (2000). PINO The Humanoid that Walk. *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2000).*

Yang, L., Chew, C. M., & Poo, A. N. (2006). Adjustable Bipedal Gait Generation using Genetic. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems October 9 - 15, 2006, Beijing, China*, 4435–4440.

Yasuda, T., Ohkura, K., & Ueda, K. (2006). A homogeneous mobile robot team that is fault-tolerant. *Advanced Engineering Informatics*. https://doi.org/10.1016/j.aei.2006.01.002

Yen, G. G., & Hickey, T. W. (2004). Reinforcement learning algorithms for robotic navigation in dynamic environments. *ISA Transactions*. https://doi.org/10.1016/s0019-0578(07)60032-9

Yokohama, P., & Takashima, T. (2002). Open Architecture Humanoid Robotics Platform. *Proceedings of the 2002 IEEE International Conference on Robotics & Automation Washington, DC,* 24–30.

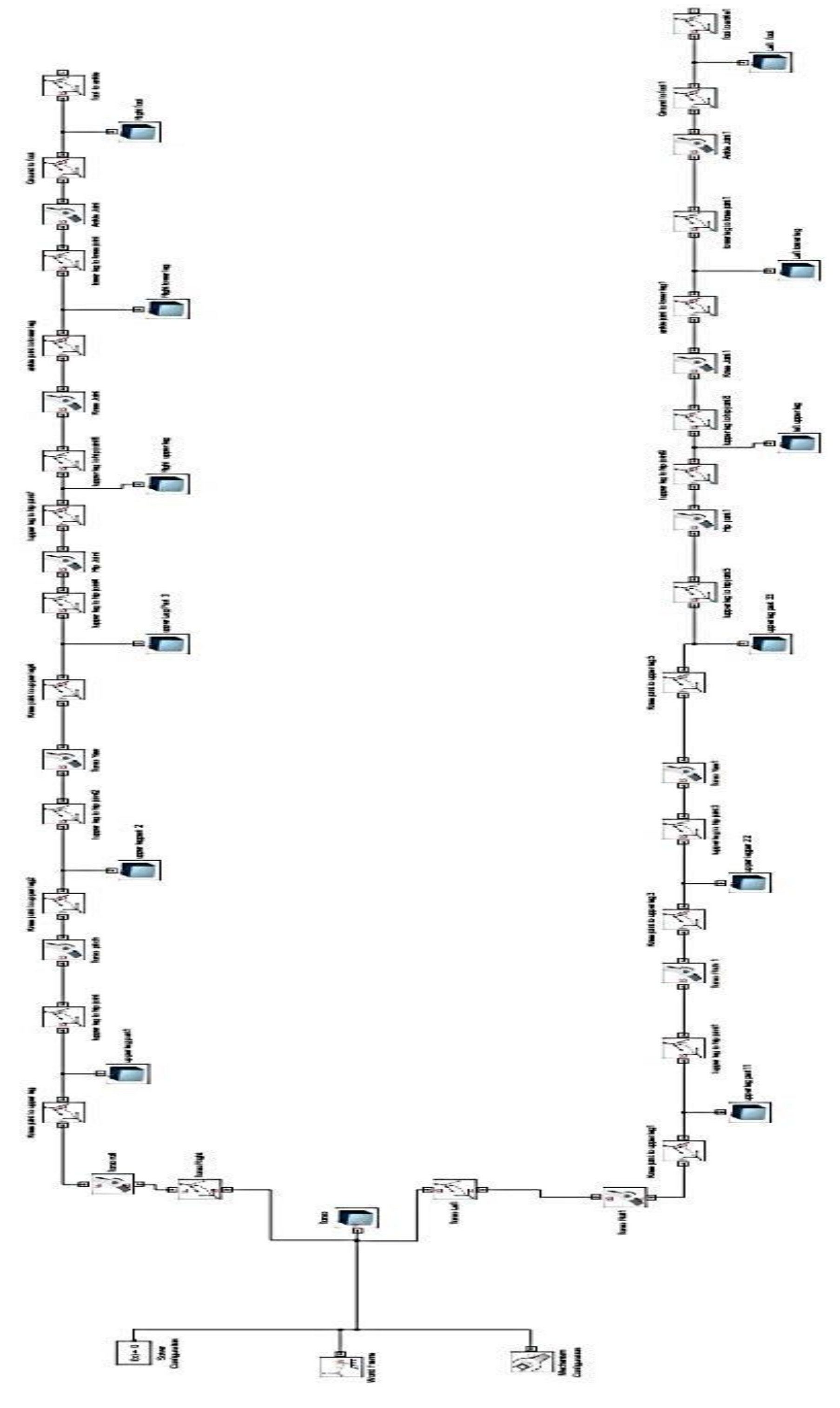
Yu, Z., Huang, Q., Ma, G., Chen, X., Zhang, W., Li, J., & Gao, J. (2014). Design and Development of the Humanoid Robot BHR-5. *Advances in Mechanical Engineering*, *2014*. https://doi.org/10.1155/2014/852937

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., & Battaglia, P. (2018). Relational deep reinforcement learning. In *arXiv*.

Zhou, C. (2002). Robot learning with GA-based fuzzy reinforcement learning agents. *Information Sciences*, *145*(1), 45–68. https://doi.org/https://doi.org/10.1016/S0020-0255(02)00223-2

APPENDIX A

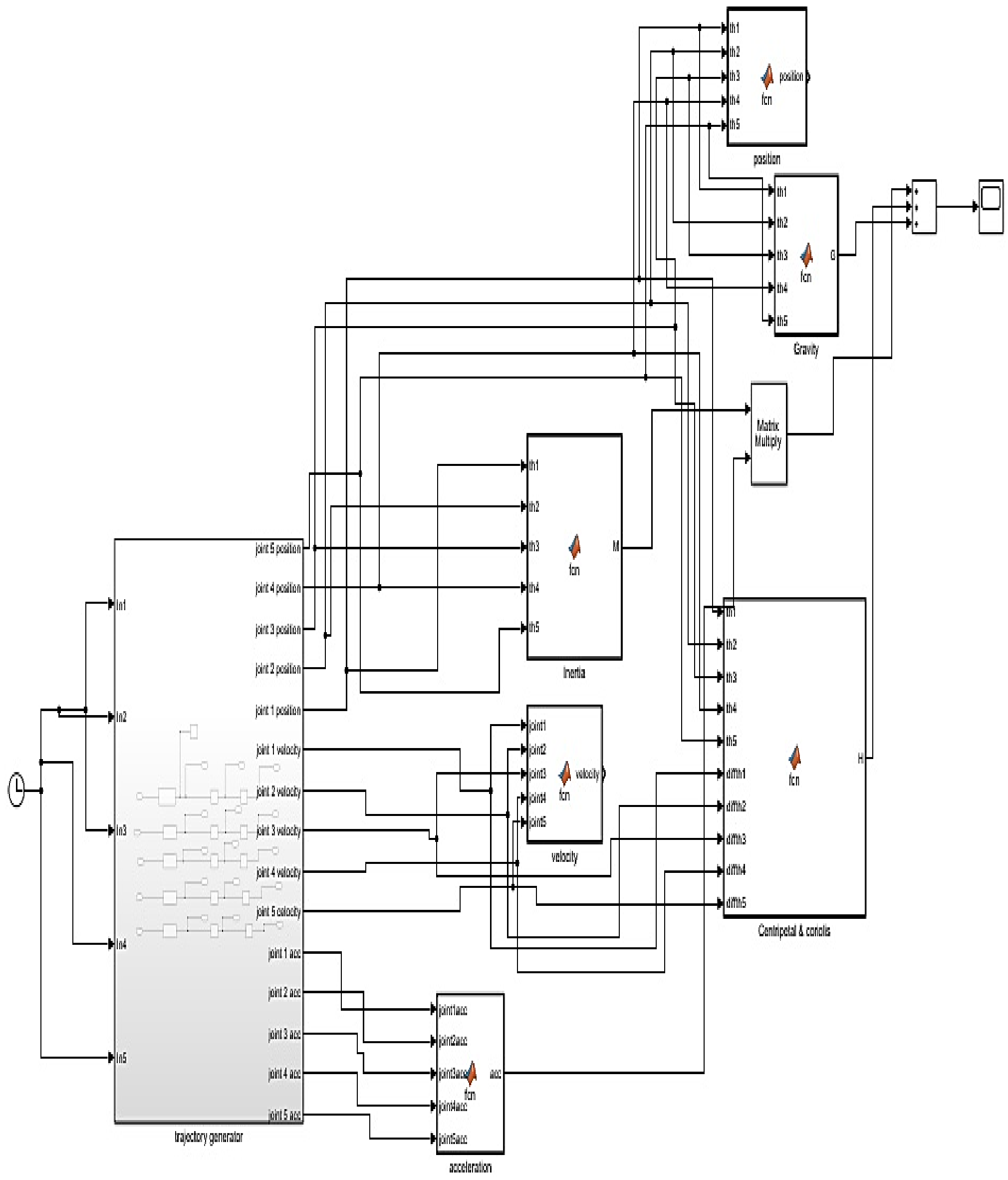
A.1 Simulink combined model of the Bipedal



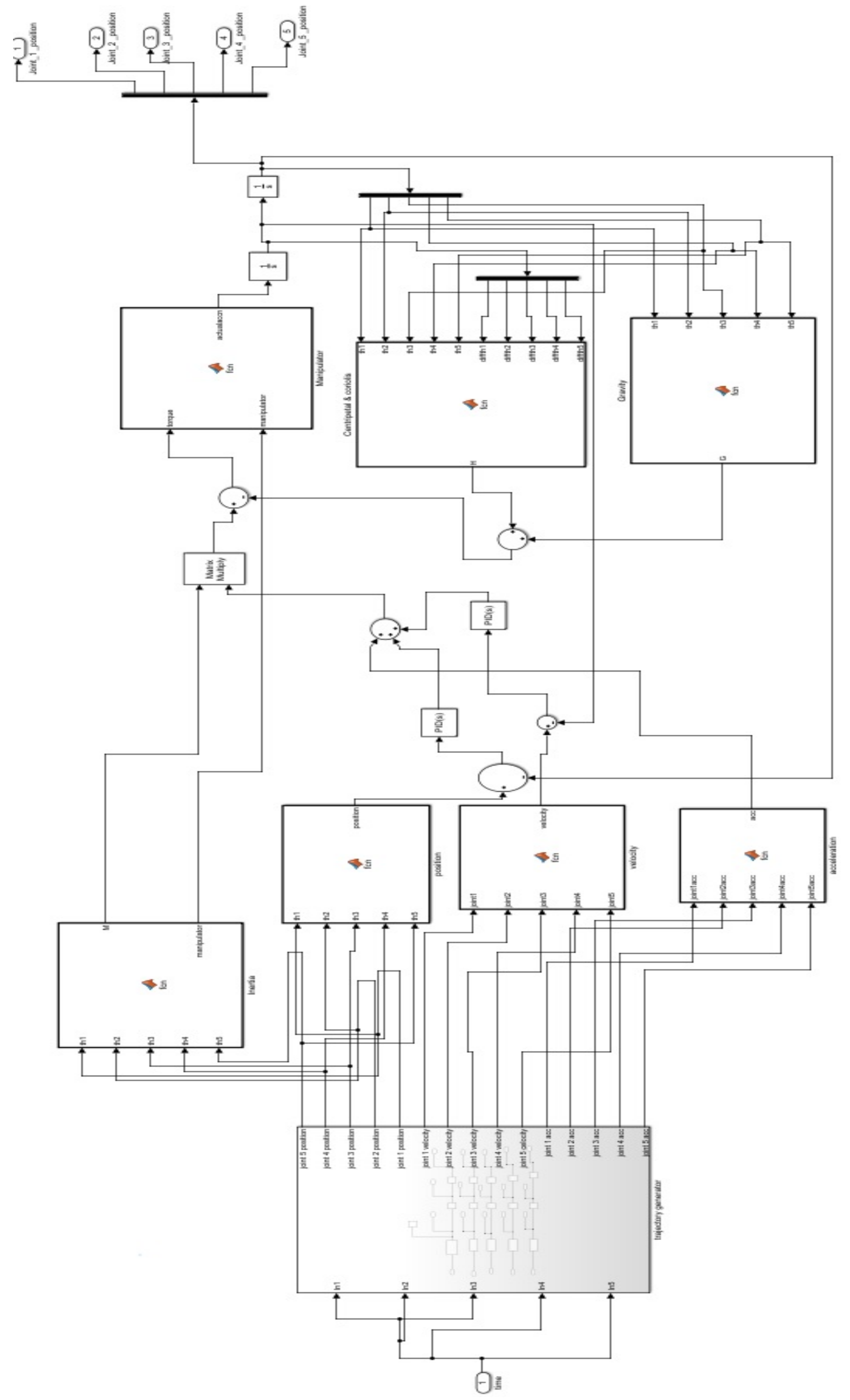
APPENDIX B

B.1 Simulink Block of Dynamic Torque for each joint

Dynamics torque is obtained for the joint trajectory of the individual joint. The maximum values of the torque are obtained for the selection of DC servomotor.

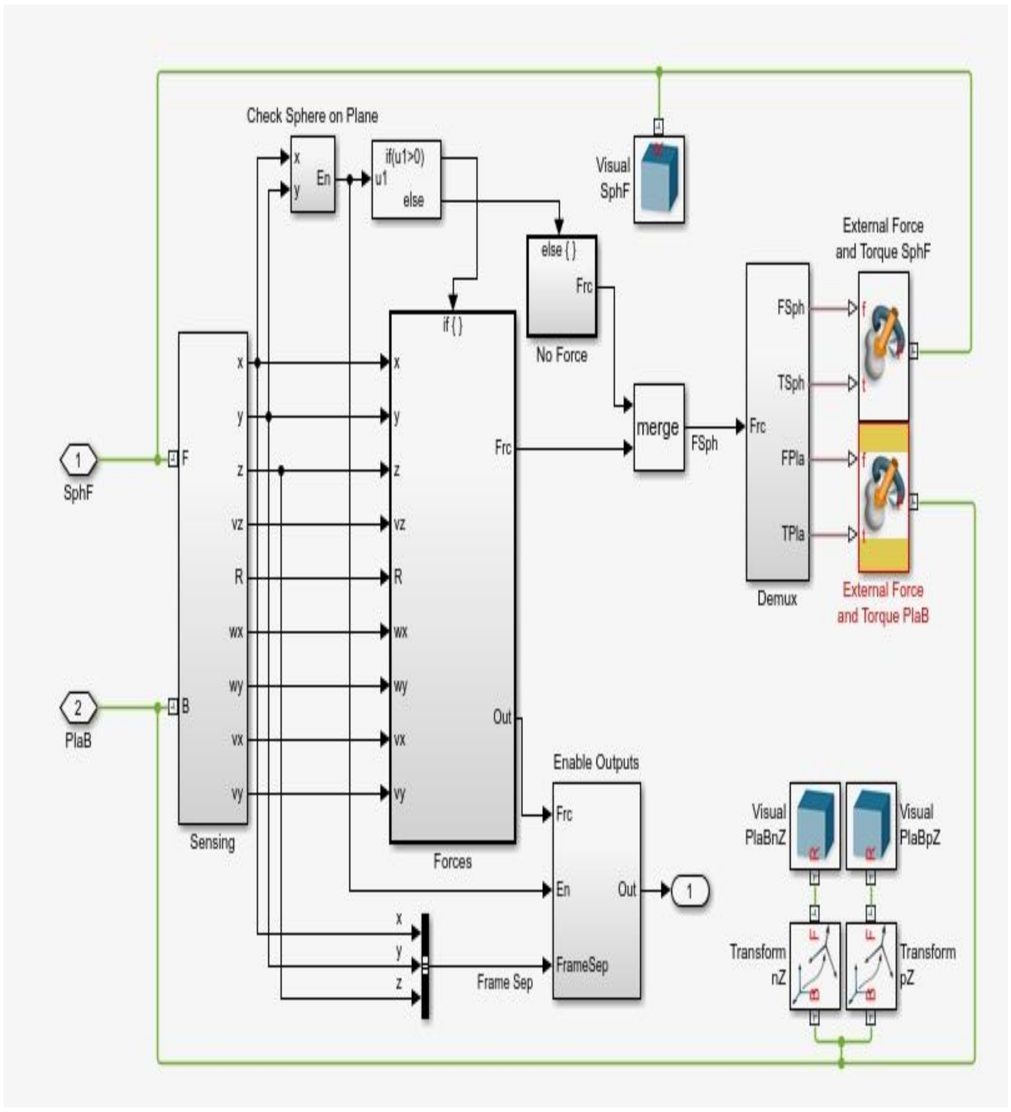


B.2 Simulink Block Diagram of Computed Torque Control

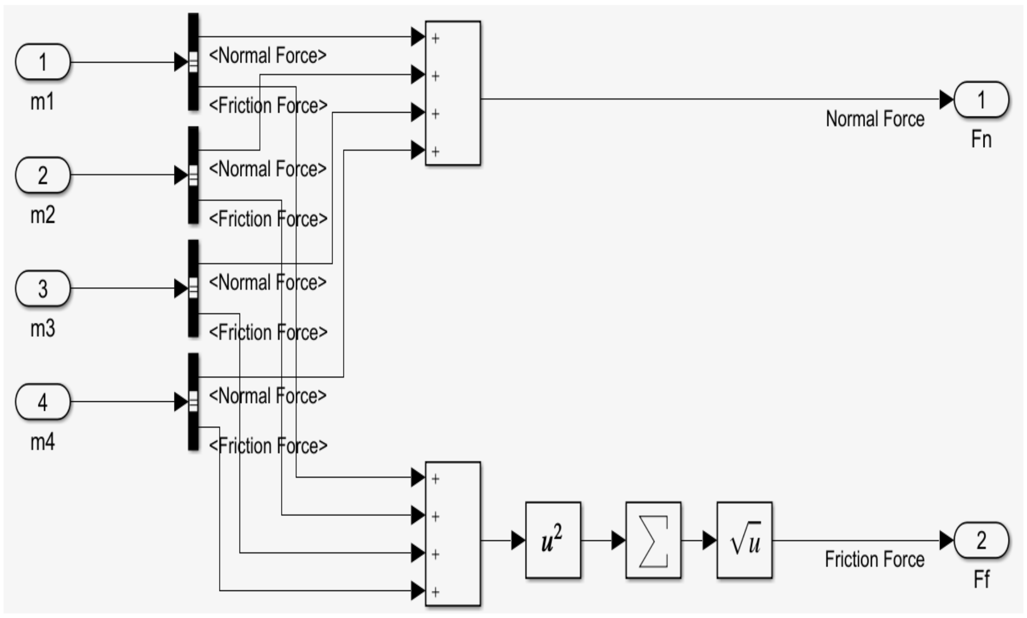


APPENDIX C

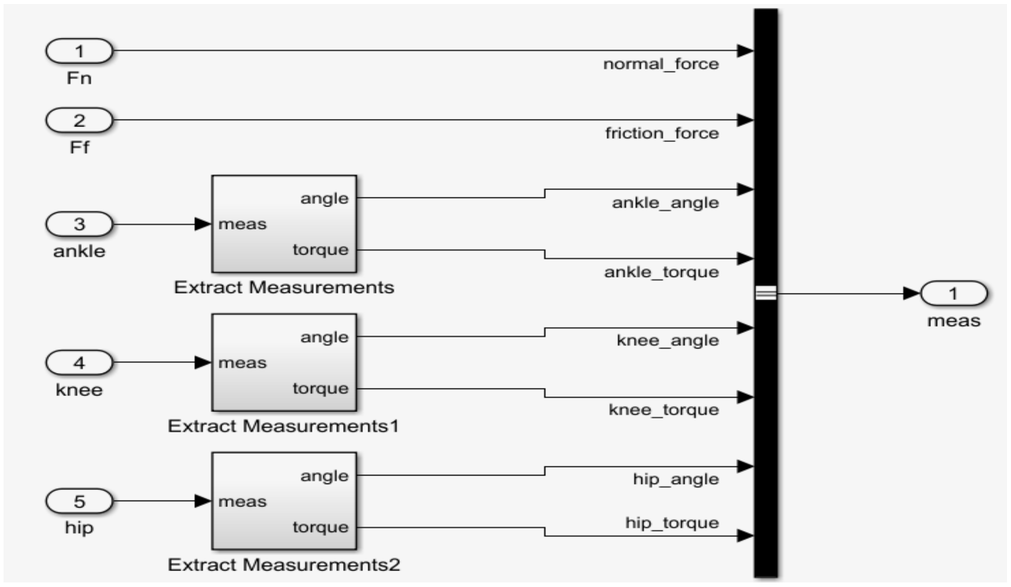
C.1 Simulink block of ground force contact

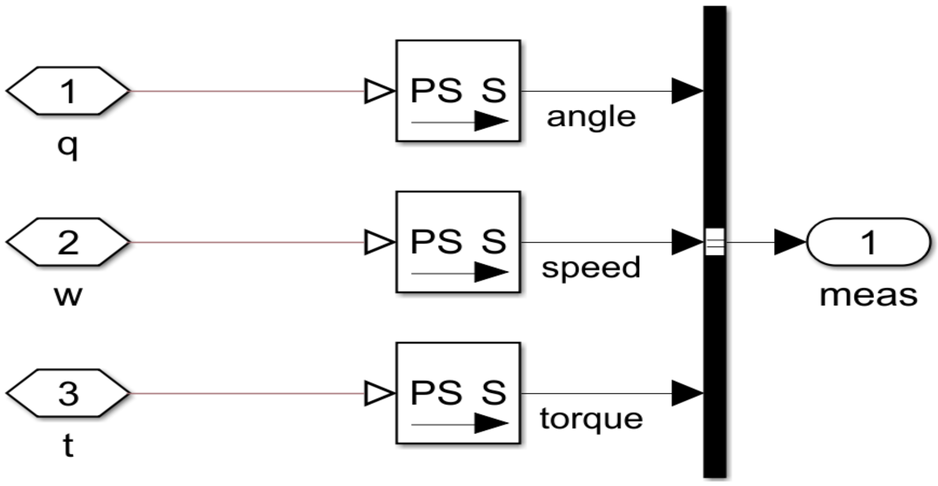


C.2 Contact Force Logging



C.3 Angle and Torque measurement for all joints




1
q
2
w
3
t
PS S
PS S
PS S
angle
speed
torque
1
meas

APPENDIX D

MATLAB CODES

bipedalparameters.m

```
%foot dimension
density = 1000; foot_x_cor = 8;foot_y_cor = 6;foot_z _cor= 1;
foot_offset_cor = [-1 0 0]; foot_density=2000;
%% Leg parameters radius, lower length, upper length
leg_rad = 0.75; low_leg_length_par = 10;up_leg_length_par = 10;
%% parameters of  Torso
torso_y = 10; torso_x = 5;torso_z = 8;torso_offset_z = -2;
torso_offset_x = -0.5; height_plane=.05;
init_height = foot_z + lower_leg_length + upper_leg_length + torso_z/2 +
                    torso_offset_z + height_plane/2;
 joint_damping = 1; joint_stiffness = 1; motion_time_constant = 0.01;
gaitPeriod = 0.8;  time = linspace(0,gaitPeriod,7)
hip_rad = deg2rad([-10, -7.5, -15, 10, 15, 10, -10]');
knee_rad = deg2rad([10, -5, 2.5, -10, -10, 15, 10]');
ankle_rad = deg2rad([-7.5 10 10 5 0 -10 -7.5]');
ankle_rad= ankle; knee_rad= knee; hip_rad= hip;
curveData = createSmoothTrajectory(ankle,knee,hip,gaitPeriod);
contact_point_radius=1e-4;  contact_stiffness = 2500;
contact_damping = 100; mu_k = 0.6; mu_s = 0.8;mu_vth = 0.1;
plane_x=25; plane_y =2.5; height_plane=.02; world_damping_value= 0.25;
world_rot_damping_value = 0.25; k_pen=5; b_penvel=3;
plane_depth = 0.025; vis_len_plaBz=plane_depth; vis_opc=1;
vis_opc_en=1;  vis_clr=[0.1 0.3 0.7]; world_damping_value = 0.25;
world_rot_damping_value = 0.25;
```

evalSmoothTrajectory.m

```
function out = evalSmoothTrajectory(params,t)
% Wrap the time value on every cycle of the trajectory
tEff = mod(t,params.gaitPeriod);
ind = find(tEff >= params.gaitTime(1:end-1));   indx = ind(end);
dt1 = tEff - params.gaitTime(indx);  out_param = zeros(3,1);
out_param(1) = params.a0_ankle(indx) + params.a1_ankle(indx)*dt 1+ ...
                    params.a2_ankle(indx)*dt1^2 + params.a3_ankle(indx)*dt1^3;
out_param(2) = params.a0_knee(indx) + params.a1_knee(indx)*dt 1+ ...
```

```
                    params.a2_knee(indx)*dt1^2 + params.a3_knee(indx)*dt1^3;
out_param(3) = params.a0_hip(indx) + params.a1_hip(indx)*dt1 + ...
                    params.a2_hip(indx)*dt1^2 + params.a3_hip(indx)*dt1^3;
end
```

createSmoothTrajectory.m

```
function curveData = createSmoothTrajectory(ankle,knee,hip,period)
%% Create necessary values for calculations
numPoints = numel(hip);   curveData.gaitPeriod = period;
curveData.gaitTime = linspace(0,period,numPoints);
dt = period/(numPoints-1);
%% Calculate derivatives .Assume zero derivatives at start and end
hip_der = [0; 0.5*( diff(hip(1:end-1)) + diff(hip(2:end)) )/dt; 0];
knee_der = [0; 0.5*( diff(knee(1:end-1)) + diff(knee(2:end)) )/dt; 0];
ankle_der = [0; 0.5*( diff(ankle(1:end-1)) + diff(ankle(2:end)) )/dt; 0];
%% Do cubic spline fitting
curveData.a0_hip = hip(1:end-1);   curveData.a1_hip = hip_der(1:end-1);
curveData.a2_hip = 3*diff(hip)/(dt^2) - 2*hip_der(1:end-1)/dt -
                              hip_der(2:end)/dt;
curveData.a3_hip = -2*diff(hip)/(dt^3) + (hip_der(1:end-1) +
                              hip_der(2:end))/(dt^2);
curveData.a0_knee = knee(1:end-1);
curveData.a1_knee = knee_der(1:end-1);
curveData.a2_knee = 3*diff(knee)/(dt^2) - 2*knee_der(1:end-1)/dt -
                              knee_der(2:end)/dt;
curveData.a3_knee = -2*diff(knee)/(dt^3) + (knee_der(1:end-1)+
                              knee_der(2:end)) / (dt^2) ;
curveData.a0_ankle = ankle(1:end-1);
curveData.a1_ankle = ankle_der(1:end-1);
curveData.a2_ankle = 3*diff(ankle)/(dt^2) - 2*ankle_der(1:end-1)/dt -
                              ankle_der (2:end) /dt;
curveData.a3_ankle = -2*diff(ankle)/(dt^3) + (ankle_der(1:end-1) +
                              ankle_der(2:end))/(dt^2);
```

plotSmoothTrajectory.m

```
% Plots cubic spline trajectory, defined by the 'curveData' variable
%% Define vectors for plots
N = 500;  t1 = linspace(0,curveData.gaitPeriod,N);
```

```
ankle_curve = zeros(size(t1)); knee_curve = zeros(size(t1));
hip_curve = zeros(size(t1));
%% Loop over all points
for indx = 1:N
trajPts = evalSmoothTrajectory(curveData,t1(indx));
ankle_curve(indx) = trajPts(1);   knee_curve(indx) = trajPts(2);
hip_curve(indx) = trajPts(3);  end
figure
subplot(3,1,1)
plot(t1,rad2deg(ankle_curve),'b-', ... curveData.gaitTime,
rad2deg([curveData.a0_ankle;curveData.a0_ankle(1)]),'ro');
title('Gait')
xlabel('Time in seconds '); ylabel('Ankle Angles in degree ');
subplot(3,1,2)
plot(t1,rad2deg(knee_curve),'b-',... curveData. gaitTime,
rad2deg([curveData.a0_knee;curveData.a0_knee(1)]),'ro');
xlabel('Time in seconds '); ylabel('Knee Angles in degree ');
subplot(3,1,3)
plot(t1,rad2deg(hip_curve),'b-', ...       curveData.gaitTime,
rad2deg([curveData.a0_hip;curveData.a0_hip(1)]),'ro');
xlabel('Time in seconds'); ylabel('Hip Angles in degree');
```

Qlearm_episodic_HIP_100.m

```
close all; clear all;  clc;
global count total_rewards total_time           % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;      % discount factor
learnRate = 0.99;    % learning rate
epsilon = 0.5;  % exploration probability(1-epsilon= exploit/epsilon = explore)
epsilonDecay = 0.98; successRate = 1;
maxEpi = 100;        % maximum number of the iterations
initialPoint = -45;   % the initial state to begin from
finalPoint = 45;     time=0;
state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];     % actions
Q = zeros(length(state),length(action));    Final_Table = table;
% main program
for episode = 1:maxEpi
% initialization
```

```
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7=0;
count=0; iter_reward =0; iter_time =0; epsilon = 0.5;
start_state = initialPoint;         goal_state = finalPoint;
startState_idx = find(state==start_state);
endState_idx= find(state==goal_state);
Imm_array=[];  disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)
tic;                  r=rand();
if (r > epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:));
cnt3=cnt3+1;   current_action = action(umax);
else
current_action=datasample(action,1); cnt4=cnt4+1;  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))
i=i+1;              if (i>=3)
next_state = state(startState_idx+1);  i=0;       end     end
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_idx)
next_state = state(startState_idx+1);
else
next_state = endState_idx; disp('goal state reached'); break;  end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
 % random reward calculation depending on current state
cnt=cnt+1;         start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
if (next_start_state_idx < startState_idx)
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx)
disp('reached goal');               cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
break;              end      cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;              actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
                    (next_reward + discount* max(Q(next_start_state_idx,:)) -
Q(startState_idx,action_idx));  cnt6=cnt6+1;
```

```
epsilon_decay(cnt6)=epsilon; epsilon = epsilon*epsilonDecay;
cnt5=cnt5+1;   reward(cnt5)=next_reward;
iter_reward = iter_reward + next_reward;      iter_time= iter_time+toc;
distance_state = endState_idx-startState_idx;
cnt2=cnt2+1;               DISTANCE(cnt2)=distance_state;
count = count+1;  random_value(count)=r;    toc       ;
Imm_array(count,:) =[r, action(action_idx), state(startState_idx),
state(next_start_state_idx), next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)
disp('epsilon <0.00001')  break;   end   end      Imm_array
header_array =["Random Number","Current Action","Current State","Next
State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100 \
          Intermediate_HIP_learning_100.xls' ,header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\HIP_100\
          Intermediate_HIP_learning_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value);
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : ');   disp(Q)
[C,I]=max(Q,[],2)  ;                        % finding the max values
disp('Q(optimal):');    disp(C);   disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;          Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1)); Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));  Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1));Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1));Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1)); Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1)); Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1)); Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(1,1));           Final_Table
Final_array{1,1} =mean_random;     Final_array{1,2}=Q;
Final_array{1,3}=C;                Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));   Final_array{1,6}=action(I(3,1));
Final_array{1,7}=action(I(4,1));   Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));   Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));  Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;     Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;  Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;         Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));  Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));  Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));  Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));  Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));  Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));  Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Header_Final_Data,episode);
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
        Final_HIP_learning_100.xls',Final_array{3},episode,'G2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{5},episode,'I2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{9},episode,'M2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{23},episode,'R2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{27},episode,'V2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{33},episode,'AB2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',action_label,episode,'A27');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',next_state_label,episode,'A30');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Final_HIP_learning_100.xls',Final_array{19},episode,'A31');
iter(episode)= count;                    total_reward(episode) = iter_reward;
total_time(episode) = iter_time;         time = time +iter_time
randomValue(episode) =mean_random;
Iteration_header_arrayValue(episode,:) =[episode, count, mean_random,
iter_reward, iter_time];  end  Iteration_header_arrayValue
 Iteration_header_array =["Episodes","Iterations","Mean Random","Total
Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning \HIP\ HIP_100\
          Graph_HIP_learning_100.xls',Iteration_header_array,1,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ HIP\ HIP_100\
          Graph_HIP_learning_100.xls',Iteration_header_arrayValue,1,'A2');
x=1:episode;   figure; plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');        ylabel('iterations');
figure; p=plot(x,randomValue,'m',x,total_reward,'b--');
p(1).LineWidth = 2;   p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location', 'northwest',
          'NumColumns', 2);
figure; plot(x,total_time,'r o-'); xlabel('Episode');     ylabel('Total Time ');
t2= datetime('now')     dt = between(t1,t2,'Time')
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required ');    disp(dt);          disp('end');
```

QLearn_episodic_KNEE_100.m

```
close all;  clear all;    clc;
global count total_rewards total_time            % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;      % discount factor
```

```
learnRate = 0.99;    % learning rate
epsilon = 0.5;  epsilonDecay = 0.98; successRate = 1;
maxEpi =100;        % maximum number of the iterations
initialPoint = 0;   % the initial state to begin from
finalPoint = 90;   time=0;
state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];     % actions
Q = zeros(length(state),length(action));     Final_Table = table;
% main program
for episode = 1:maxEpi
% initialization
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7=0;
count=0;i=0;   iter_reward =0; iter_time =0; epsilon = 0.5;
start_state = initialPoint;        goal_state = finalPoint;
startState_idx = find(state==start_state);
endState_idx= find(state==goal_state);
Imm_array=[];  disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)  tic;               r=rand() ;
if (r>epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:)); disp('in Exploit'); cnt3=cnt3+1;
current_action = action(umax);
else      current_action=datasample(action,1);
cnt4=cnt4+1;   disp('in Explore');  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
disp('startState_idx <= endState_idx-1 &&   startState_idx>=initialPoint');
          next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))
disp('next_state ==state(startState_idx)');      i=i+1;
if (i>=3)
next_state = state(startState_idx+1);  disp('in same state for 3 iterations ');
i=0;      end        end
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_idx)
disp('startState_idx >= endState_idx-1 && startState_idx < endState_idx-1')
next_state = state(startState_idx+1);
else
next_state = endState_idx; disp('goal state reached'); break;   end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
 % random reward calculation depending on current state
cnt=cnt+1;         start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
```

```
if (next_start_state_idx < startState_idx)
disp('in if next_start_state < startState_idx');
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx)  disp('reached goal');
cnt1=cnt1+1;   next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
break;  end                    cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;   actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
(next_reward + discount* max(Q(next_start_state_idx,:)) -
          Q(startState_idx,action_idx));
cnt6=cnt6+1;   epsilon_decay(cnt6)=epsilon; epsilon = epsilon*epsilonDecay ;
cnt5=cnt5+1;   reward(cnt5)=next_reward;
iter_reward = iter_reward + next_reward;      iter_time= iter_time+toc;
distance_state = endState_idx-startState_idx;           cnt2=cnt2+1;
DISTANCE(cnt2)=distance_state;     count = count+1;
random_value(count)=r;         toc       ;
Imm_array(count,:) =[r, action(action_idx), state(startState_idx),
state(next_start_state_idx),next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)
disp('epsilon <0.00001')  break;         end  end
 header_array =["Random Number","Current Action","Current State","Next
State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
          Intermediate_KNEE_learning_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
          Intermediate_KNEE_learning_100.xls',Imm_array,episode,'A2');
Imm_array
header_array =["Random Number","Current Action","Current State","Next
State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
          Intermediate_KNEE_learning_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
          Intermediate_KNEE_learning_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value)
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : ');   disp(Q)
```

```
[C,I]=max(Q,[],2)  ;                        % finding the max values
disp('Q(optimal):');  disp(C);  disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;           Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1));  Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));  Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1));Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1));Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1));Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1));Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1));Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(21,1));         Final_Table
Final_array{1,1} =mean_random;     Final_array{1,2}=Q;
Final_array{1,3}=C;   Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));       Final_array{1,6}=action(I(3,1));
Final_array{1,7}=action(I(4,1));       Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));       Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));      Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;         Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;   Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;          Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));   Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));   Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));   Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));   Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));   Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));   Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Header_Final_Data,episode);
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{3},episode,'G2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{5},episode,'I2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{9},episode,'M2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{23},episode,'R2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{27},episode,'V2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{33},episode,'AB2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',action_label,episode,'A27');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',next_state_label,episode,'A30');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Final_KNEE_learning_100.xls',Final_array{19},episode,'A31');
iter(episode)= count;            total_reward(episode) = iter_reward;
total_time(episode) = iter_time;  time = time +iter_time
randomValue(episode) =mean_random;
Iteration_header_arrayValue(episode,:)
                        =[episode,count,mean_random,iter_reward,iter_time];
end     Iteration_header_arrayValue
Iteration_header_array =["Episodes","Iterations","Mean Random","Total
Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Graph_KNEE_learning_100.xls',Iteration_header_array,1,'A1');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ KNEE\ KNEE_100\
        Graph_KNEE_learning_100.xls',Iteration_header_arrayValue,1,'A2');
x=1:episode;   figure; plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');      ylabel('iterations');     figure;
p=plot(x,randomValue,'m',x,total_reward,'b--');  p(1).LineWidth = 2;
p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location',
                        'northwest','NumColumns',2);
figure; plot(x,total_time,'r o-'); xlabel('Episode');     ylabel('Total Time ');
t2= datetime('now')    dt = between(t1,t2,'Time')
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required ');   disp(dt);         disp('end');
```

QLearn_episodic_ANKLE_100.m

```
close all; clear all;clc;
global count total_rewards total_time          % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;     % discount factor
learnRate = 0.99;   % learning rate
epsilon = 0.5;  epsilonDecay = 0.98; successRate = 1;
maxEpi =100;       % maximum number of the iterations
initialPoint = -30  % the initial state to begin from
finalPoint = 30;         time=0;
state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];    % actions
Q = zeros(length(state),length(action));   Final_Table = table;
% main program
for episode = 1:maxEpi
epiStartTime = datetime('now');
% initialization
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7=0;count=0;i=0;
iter_reward =0;  iter_time =0;epsilon = 0.5;
start_state = initialPoint; goal_state = finalPoint;
startState_idx = find(state==start_state);
endState_idx= find(state==goal_state);        Imm_array=[];
disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)  tic;    r=rand();
if (r>epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:));
```

```
cnt3=cnt3+1;   current_action = action(umax);
else
current_action=datasample(action,1); cnt4=cnt4+1;  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))
i=i+1;  if (i>=3)  next_state = state(startState_idx+1);
i=0;                end                end
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_idx)
next_state = state(startState_idx+1);  else
next_state = endState_idx;  disp('goal state reached'); break;  end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
% random reward calculation depending on current state
cnt=cnt+1;         start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
if (next_start_state_idx < startState_idx)
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx)
disp('reached goal');              cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
break;  end                cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;               actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
                    (next_reward + discount* max(Q(next_start_state_idx,:)) -
                    Q(startState_idx,action_idx));
cnt6=cnt6+1;               epsilon_decay(cnt6)=epsilon;
epsilon = epsilon*epsilonDecay ;       cnt5=cnt5+1;
reward(cnt5)=next_reward;    iter_reward = iter_reward + next_reward;
iter_time= iter_time+toc;        distance_state = endState_idx-startState_idx;
cnt2=cnt2+1;               ISTANCE(cnt2)=distance_state;
count = count+1;         random_value(count)=r;         toc       ;
Imm_array(count,:) =[r, action(action_idx), state(startState_idx),
state(next_start_state_idx), next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)
disp('epsilon <0.00001') break;     end  end    Imm_array
```

```
header_array =["Random Number","Current Action","Current State","Next
State", "Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Intermediate_ANKLE_learning_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Intermediate_ANKLE_learning_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value)
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : ');        disp(Q)
[C,I]=max(Q,[],2)  ;                        % finding the max values
disp('Q(optimal):');     disp(C);          disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;          Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1));  Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));  Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1));Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1));Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1));Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1));Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1));Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(21,1));         Final_Table
Final_array{1,1} =mean_random;     Final_array{1,2}=Q;
Final_array{1,3}=C;   Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));       Final_array{1,6}=action(I(3,1));
Final_array{1,7}=action(I(4,1));       Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));       Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));      Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;         Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;   Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
```

```
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;          Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));   Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));   Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));   Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));   Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));   Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));   Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Header_Final_Data,episode);
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{3},episode,'G2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{5},episode,'I2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
        Final_ANKLE_learning_100.xls',Final_array{9},episode,'M2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{23},episode,'R2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{27},episode,'V2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{33},episode,'AB2');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',action_label,episode,'A27');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',next_state_label,episode,'A30')
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Final_ANKLE_learning_100.xls',Final_array{19},episode,'A31');
```

```
iter(episode)= count;  randomValue(episode) =mean_random;
total_reward(episode) = iter_reward;
total_time(episode) = iter_time; time = time +iter_time
Iteration_header_arrayValue(episode,:) =[episode, count, mean_random,
                    iter_reward, iter_time];
end Iteration_header_arrayValue
 Iteration_header_array =["Episodes","Iterations","Mean Random","Total
Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
          Graph_ANKLE_learning_100.xls',Iteration_header_array,1,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal learning\ ANKLE\ ANKLE_100\
       Graph_ANKLE_learning_100.xls',Iteration_header_arrayValue,1,'A2');
x=1:episode;   figure; plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');        ylabel('iterations');      figure;
p=plot(x,randomValue,'m',x,total_reward,'b--');
p(1).LineWidth = 2;   p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location', 'northwest',
                    'NumColumns',2);
 figure; plot(x,total_time,'r o-'); xlabel('Episode');     ylabel('Total Time ');
t2= datetime('now');    dt = between(t1,t2,'Time');
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required execute and plot values ');  disp(dt); disp('end');
```

QLearn_execution_HIP_100.m

```
close all; clear all; clc;
global count total_rewards total_time          % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;      % discount factor
learnRate = 0.99;    % learning rate
epsilon = 0.5;  epsilonDecay = 0.98; successRate = 1;
maxEpi = 100;        % maximum number of the iterations
initialPoint = -45;   % the initial state to begin from
finalPoint = 45;          time=0;
state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];     % actions
Q = zeros(length(state),length(action));    Final_Table = table;
% main program
for episode = 1:maxEpi
```

```
% initialization
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7=0;count=0;
iter_reward =0;  iter_time =0; epsilon = 0.5;  size_state =size(state(2:19),2);
rand_State = randsample(size_state,1)
start_state = state(rand_State) ; goal_state = finalPoint;
startState_idx = find(state==start_state);
endState_idx= find(state==goal_state);
Imm_array=[];  disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)  tic;     r=rand();
if (r>epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:)); cnt3=cnt3+1;
current_action = action(umax);
else
current_action=datasample(action,1); cnt4=cnt4+1;  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))
i=i+1;  if (i>=3) next_state = state(startState_idx+1);  i=0;    end  end
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_idx)
next_state = state(startState_idx+1);
else
next_state = endState_idx;  disp('goal state reached');           break;  end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
cnt=cnt+1;   start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
if (next_start_state_idx < startState_idx)
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx)
disp('reached goal');               cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);break;  end
cnt1=cnt1+1;   next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;              actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
(next_reward + discount* max(Q(next_start_state_idx,:)) -
Q(startState_idx,action_idx)); cnt6=cnt6+1;  epsilon_decay(cnt6)=epsilon;
epsilon = epsilon*epsilonDecay; cnt5=cnt5+1; reward(cnt5)=next_reward;
iter_reward = iter_reward + next_reward;  iter_time= iter_time+toc;
```

```
distance_state = endState_idx-startState_idx; cnt2=cnt2+1;
DISTANCE(cnt2)=distance_state;     count = count+1;
random_value(count)=r;         toc       ;
Imm_array(count,:) =[r, action(action_idx), state(startState_idx),
state(next_start_state_idx), next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)  disp('epsilon <0.00001') break; end    end      Imm_array
header_array =["Random Number","Current Action","Current State","Next
          State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Intermediate_HIP_execution_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Intermediate_HIP_execution_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value);
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : ');         disp(Q)
[C,I]=max(Q,[],2)  ;                         % finding the max values
disp('Q(optimal):');     disp(C);          disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;          Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1));  Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));  Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1));Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1));Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1));Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1));Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1));Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(21,1));         Final_Table
Final_array{1,1} =mean_random;     Final_array{1,2}=Q;
Final_array{1,3}=C;   Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));       Final_array{1,6}=action(I(3,1));
```

```
Final_array{1,7}=action(I(4,1));        Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));        Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));       Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;          Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;   Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));   Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));   Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));   Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));   Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));   Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));   Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Header_Final_Data,episode);
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{3},episode,'G2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
         Final_HIP_execution_100.xls',Final_array{5},episode,'I2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{9},episode,'M2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{23},episode,'R2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{27},episode,'V2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100 \
          Final_HIP_execution_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{33},episode,'AB2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',action_label,episode,'A27');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',next_state_label,episode,'A30');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Final_HIP_execution_100.xls',Final_array{19},episode,'A31');
iter(episode)= count;   total_reward(episode) = iter_reward;
total_time(episode) = iter_time;        time = time +iter_time
randomValue(episode) =mean_random;
Iteration_header_arrayValue(episode,:) =[episode, count, mean_random,
iter_reward, iter_time];        end             Iteration_header_arrayValue
Iteration_header_array =["Episodes","Iterations","Mean Random","Total
          Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Graph_HIP_execution_100.xls',Iteration_header_array,1,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ HIP\ HIP_100\
          Graph_HIP_execution_100.xls',Iteration_header_arrayValue,1,'A2');
x=1:episode;   figure; plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');      ylabel('iterations');      figure;
p=plot(x,randomValue,'m',x,total_reward,'b--');      p(1).LineWidth = 2;
p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location', 'northwest',
          'NumColumns',2);
figure; plot(x,total_time,'r o-');  xlabel('Episode');    ylabel('Total Time ');
t2= datetime('now');    dt = between(t1,t2,'Time')
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required ');    disp(dt);              disp('end');
```

QLearn_Execution_KNEE_100.m

```
close all;        clear all;        clc;
global count total_rewards total_time          % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;      % discount factor
learnRate = 0.99;    % learning rate
epsilon = 0.5;   epsilonDecay = 0.98;           successRate = 1;
maxEpi =100;        % maximum number of the iterations
initialPoint = 0;   % the initial state to begin from
```

```
finalPoint = 90;    % the final state to reach to
time=0;  state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];     % actions
Q = zeros(length(state),length(action));    Final_Table = table;
% main program
for episode = 1:maxEpi
% initialization
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7= count=0;i=0;
iter_reward =0; iter_time =0; epsilon = 0.5;  size_state =size(state(2:19),2);
rand_State = randsample(size_state,1); start_state = state(rand_State);
goal_state = finalPoint;          startState_idx = find(state==start_state);
endState_idx= find(state==goal_state); Imm_array=[];
disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)  tic;                r=rand() ;
if (r>epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:));   disp('in Exploit');        cnt3=cnt3+1;
current_action = action(umax);
else
current_action=datasample(action,1); cnt4=cnt4+1;  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
disp('startState_idx <= endState_idx-1 && startState_idx>=initialPoint');
next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))
disp('next_state ==state(startState_idx)');       i=i+1;  if (i>=3)
next_state = state(startState_idx+1);  disp('in same state for 3 iterations ');
i=0;                 end                end
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_id
disp('startState_idx >= endState_idx-1 && startState_idx < endState_idx-1')
next_state = state(startState_idx+1);
else
next_state = endState_idx; disp('goal state reached'); break; end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
cnt=cnt+1;   start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
if (next_start_state_idx < startState_idx)
disp('in if next_start_state < startState_idx');
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx) disp('reached
goal');cnt1=cnt1+1; next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
```

```
break;  end      cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;              actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
                    (next_reward + discount* max(Q(next_start_state_idx,:)) -
                    Q(startState_idx,action_idx));
cnt6=cnt6+1;   epsilon_decay(cnt6)=epsilon;
epsilon = epsilon*epsilonDecay ;cnt5=cnt5+1;reward(cnt5)=next_reward;
iter_reward = iter_reward + next_reward;      iter_time= iter_time+toc;
distance_state = endState_idx-startState_idx; cnt2=cnt2+1;
DISTANCE(cnt2)=distance_state;     count = count+1;
random_value(count)=r;                  toc;
Imm_array(count,:) =[r,action(action_idx), state(startState_idx),
state(next_start_state_idx), next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)  disp('epsilon <0.00001') break; end              end
header_array =["Random Number","Current Action","Current State","Next
          State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Intermediate_KNEE_execution_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\
KNEE_100\Intermediate_KNEE_execution_100.xls',Imm_array,episode,'A2');
Imm_array
header_array =["Random Number","Current Action","Current State","Next
                    State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Intermediate_KNEE_execution_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Intermediate_KNEE_execution_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value)
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : '); disp(Q);[C,I]=max(Q,[],2) ; % finding the max values
disp('Q(optimal):');                disp(C);           disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;           Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1));  Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1)); Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1)); Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1)); Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1)); Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1)); Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(21,1));        Final_Table
Final_array{1,1} =mean_random;     Final_array{1,2}=Q;
Final_array{1,3}=C;                Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));   Final_array{1,6}=action(I(3,1));
Final_array{1,7}=action(I(4,1));   Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));   Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));  Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;     Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;  Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;         Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));  Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));  Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));  Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));  Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));  Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));  Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
        Final_KNEE_execution_100.xls',Header_Final_Data,episode);
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{3},episode,'G2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{5},episode,'I2')
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{9},episode,'M2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{23},episode,'R2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution \ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{27},episode,'V2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{33},episode,'AB2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',action_label,episode,'A27');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',next_state_label,episode,'A30');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Final_KNEE_execution_100.xls',Final_array{19},episode,'A31');
iter(episode)= count;  total_reward(episode) = iter_reward;
total_time(episode) = iter_time; time = time +iter_time
randomValue(episode) =mean_random;
Iteration_header_arrayValue(episode,:)
          =[episode,count,mean_random,iter_reward,iter_time];
end                Iteration_header_arrayValue
Iteration_header_array =["Episodes","Iterations","Mean Random","Total
Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Graph_KNEE_execution_100.xls',Iteration_header_array,1,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ KNEE\ KNEE_100\
          Graph_KNEE_execution_100.xls',Iteration_header_arrayValue,1,'A2');
=1:episode;     figure; plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');                 ylabel('iterations');
figure; p=plot(x,randomValue,'m',x,total_reward,'b--');
p(1).LineWidth = 2;              p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location', 'northwest',
          'NumColumns',2);
```

```
figure; plot(x,total_time,'r o-'); xlabel('Episode');      ylabel('Total Time ');
t2= datetime('now');               dt = between(t1,t2,'Time')
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required ');              disp(dt);                   disp('end');
```

QLearn_Execution_ANKLE_100.m

```
close all;          clear all;          clc;
global count total_rewards total_time            % global parameters
t1= datetime('now')
% learning parameters
discount = 0.9;      % discount factor
learnRate = 0.99;    % learning rate
epsilon = 0.5;  epsilonDecay = 0.98; successRate = 1;
maxEpi =100;        % maximum number of the iterations
initialPoint = -30  % the initial state to begin from
finalPoint = 30;           time=0;
state =linspace(initialPoint,finalPoint,21) % state
action = [0,1];     % actions
Q = zeros(length(state),length(action));    Final_Table = table;
% main program
for episode = 1:maxEpi
epiStartTime = datetime('now');
% initialization
cnt=0;cnt1=0; cnt2=0; cnt3=0;cnt4=0;cnt5=0;cnt6=0;cnt7=0;count=0;i=0;
iter_reward =0; iter_time =0; epsilon = 0.5; size_state =size(state(2:19),2);
rand_State = randsample(size_state,1)
start_state = state(rand_State);           goal_state = finalPoint;
startState_idx = find(state==start_state);
endState_idx= find(state==goal_state);
Imm_array=[];            disp(['Episode: ' num2str(episode)]);
while(start_state < goal_state)  tic;                          r=rand();
if (r>epsilon || episode == maxEpi) && r<=successRate
[~,umax]=max(Q(startState_idx,:));   cnt3=cnt3+1;
current_action = action(umax);
else
current_action=datasample(action,1); cnt4=cnt4+1;  end
action_idx = find(action==current_action);
if (startState_idx <= endState_idx-1 && startState_idx>=initialPoint)
next_state = state(startState_idx+current_action) ;
if (next_state ==state(startState_idx))           i=i+1;  if (i>=3)
next_state = state(startState_idx+1);  i=0;  end  end
```

```
elseif (startState_idx >= endState_idx-1 && startState_idx < endState_idx)
next_state = state(startState_idx+1);
else
next_state = endState_idx;  disp('goal state reached'); break;end
next_start_state_idx = find(state==next_state);
next_reward = exp(-learnRate*(endState_idx-startState_idx));
cnt=cnt+1;      start_stateARR(cnt)=startState_idx;
start_stateARR_Value(cnt)=state(startState_idx);
if (next_start_state_idx < startState_idx)
next_start_state_idx = startState_idx+1;
elseif(next_start_state_idx ==  endState_idx)  disp('reached goal');
cnt1=cnt1+1;   next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
break;  end      cnt1=cnt1+1;
next_start_stateARR(cnt1)=next_start_state_idx;
next_start_stateARR_value(cnt1)=state(next_start_state_idx);
cnt7=cnt7+1;              actionARR(cnt7)=action_idx;
actionARR_value(cnt7)=action(action_idx);
Q(startState_idx,action_idx) = Q(startState_idx,action_idx) + learnRate *
                    (next_reward + discount* max(Q(next_start_state_idx,:)) -
                    Q(startState_idx,action_idx));
cnt6=cnt6+1;              epsilon_decay(cnt6)=epsilon;
epsilon = epsilon*epsilonDecay ;  cnt5=cnt5+1; reward(cnt5)=next_reward;
iter_reward = iter_reward + next_reward;
iter_time= iter_time+toc;       distance_state = endState_idx-startState_idx;
cnt2=cnt2+1;   DISTANCE(cnt2)=distance_state;
count = count+1;         random_value(count)=r;         toc       ;
Imm_array(count,:) =[r, action(action_idx), state(startState_idx),
state(next_start_state_idx),next_reward,epsilon,iter_time,distance_state];
startState_idx = next_start_state_idx;
if (epsilon <0.00001)  disp('epsilon <0.00001') break; end    end Imm_array
header_array =["Random Number","Current Action","Current State","Next
          State","Reward","Epsilon","Time","Distance"];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Intermediate_ANKLE_execution_100.xls',header_array,episode,'A1');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Intermediate_ANKLE_execution_100.xls',Imm_array,episode,'A2');
mean_random = mean(random_value)
disp(['Total Iteration:   ' num2str(count) '   Total exploit:   ' num2str(cnt3) '
Total explore:   ' num2str(cnt4)]);
disp('Final Q matrix : '); disp(Q);[C,I]=max(Q,[],2);  % finding the max values
disp('Q(optimal):');              disp(C);          disp('Optimal Policy');
disp([action(I(1,1)) action(I(2,1)) action(I(3,1)) action(I(4,1)) action(I(5,1))
action(I(6,1)) action(I(7,1)) action(I(8,1)) action(I(9,1)) action(I(10,1))
action(I(11,1)) action(I(12,1)) action(I(13,1)) action(I(14,1)) action(I(15,1))
action(I(16,1)) action(I(17,1)) action(I(18,1)) action(I(19,1)) action(I(20,1))
action(I(21,1))]);
Final_Table.Episode=episode;                    Final_Table.Iteration=count;
Final_Table.Mean_Random=mean_random;
Final_Table.Total_Rewards = iter_reward;
Final_Table.Total_Time= iter_time;
Final_Table.OptPol1=action(I(1,1));  Final_Table.OptPol2=action(I(2,1));
Final_Table.OptPol3=action(I(3,1));  Final_Table.OptPol4=action(I(4,1));
Final_Table.OptPol5=action(I(5,1));  Final_Table.OptPol6=action(I(6,1));
Final_Table.OptPol7=action(I(7,1));  Final_Table.OptPol8=action(I(8,1));
Final_Table.OptPol9=action(I(9,1));  Final_Table.OptPol10=action(I(10,1));
Final_Table.OptPol11=action(I(11,1));Final_Table.OptPol12=action(I(12,1));
Final_Table.OptPol13=action(I(13,1));Final_Table.OptPol14=action(I(14,1));
Final_Table.OptPol15=action(I(15,1));Final_Table.OptPol16=action(I(16,1));
Final_Table.OptPol17=action(I(17,1));Final_Table.OptPol18=action(I(18,1));
Final_Table.OptPol19=action(I(19,1));Final_Table.OptPol20=action(I(20,1));
Final_Table.OptPol21=action(I(21,1));         Final_Table
Final_array{1,1} =mean_random;              Final_array{1,2}=Q;
Final_array{1,3}=C;                         Final_array{1,4}=action(I(1,1));
Final_array{1,5}=action(I(2,1));            Final_array{1,6}=action(I(3,1));
Final_array{1,7}=action(I(4,1));            Final_array{1,8}=action(I(5,1));
Final_array{1,9}=action(I(6,1));            Final_array{1,10}=action(I(7,1));
Final_array{1,11}=action(I(8,1));           Final_array{1,12}=action(I(9,1));
Final_array{1,13}=iter_reward;              Final_array{1,14}=iter_time;
Final_array{1,15}=start_stateARR;           Final_array{1,16}=actionARR;
Final_array{1,17}=next_start_stateARR;
Final_array{1,18}=start_stateARR_Value;
Final_array{1,19}=next_start_stateARR_value;
Final_array{1,20}=actionARR_value;          Final_array{1,21} =count;
Final_array{1,22} =action(I(10,1));   Final_array{1,23} =action(I(11,1));
Final_array{1,24} =action(I(12,1));   Final_array{1,25} =action(I(13,1));
Final_array{1,26} =action(I(14,1));   Final_array{1,27} =action(I(15,1));
Final_array{1,28} =action(I(16,1));   Final_array{1,29} =action(I(17,1));
Final_array{1,30} =action(I(18,1));   Final_array{1,31} =action(I(19,1));
Final_array{1,32} =action(I(20,1));   Final_array{1,33} =action(I(21,1));
Header_Final_Data(1,:) =["Iterations","Mean Random","Total Reward","Total
Time","Q(:,1)","Q(:,2)","C","Action 1","Action 2","Action 3","Action
4","Action 5","Action 6","Action 7","Action 8","Action 9","Action
10","Action 11","Action 12","Action 13","Action 14","Action 15","Action
16","Action 17","Action 18","Action 19","Action 20","Action 21"];
start_state_label(1,:) =["Start State :",""];
action_label(1,:) =["Action Taken :",""];
next_state_label(1,:) =["Next State :",""];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ANKLE_100\
        Final_ANKLE_execution_100.xls',Header_Final_Data,episode);
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{21},episode,'A2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{1},episode,'B2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{13},episode,'C2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{14},episode,'D2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{2},episode,'E2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{3},episode,'G2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{4},episode,'H2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{5},episode,'I2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{6},episode,'J2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{7},episode,'K2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{8},episode,'L2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{9},episode,'M2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{10},episode,'N2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{11},episode,'O2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{12},episode,'P2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{22},episode,'Q2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
        Final_ANKLE_execution_100.xls',Final_array{23},episode,'R2');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{24},episode,'S2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{25},episode,'T2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{26},episode,'U2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{27},episode,'V2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{28},episode,'W2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{29},episode,'X2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{30},episode,'Y2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{31},episode,'Z2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{32},episode,'AA2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{33},episode,'AB2');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',start_state_label,episode,'A24');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',action_label,episode,'A27');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',next_state_label,episode,'A30');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{18},episode,'A25');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{20},episode,'A28');
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Final_ANKLE_execution_100.xls',Final_array{19},episode,'A31');
iter(episode)= count; randomValue(episode) =mean_random;
total_reward(episode) = iter_reward; total_time(episode) = iter_time;
time = time +iter_time
Iteration_header_arrayValue(episode,:) =[episode, count, mean_random,
                    iter_reward, iter_time];
end                 Iteration_header_arrayValue
 Iteration_header_array =["Episodes","Iterations","Mean Random","Total
                    Rewards", "Total Time" ];
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
          Graph_ANKLE_execution_100.xls',Iteration_header_array,1,'A1');
```

```
xlswrite('C:\Users\rashmi\Desktop\bipedal execution\ ANKLE\ ANKLE_100\
Graph_ANKLE_execution_100.xls',Iteration_header_arrayValue,1,'A2');
x=1:episode;   figure;  plot(x, Iteration_header_arrayValue(:,2),'k o-');
xlabel('Episode');       ylabel('iterations');               figure;
p=plot(x,randomValue,'m',x,total_reward,'b--');       p(1).LineWidth =
2;p(2).Marker = '*';
legend({'Random Values', 'Total Reward'}, 'Location', 'northwest',
          'NumColumns',2);
figure;  plot(x,total_time,'r o-');   xlabel('Episode');  ylabel('Total Time ');
t2= datetime('now');              dt = between(t1,t2,'Time');
disp(['Total number Episode: ' num2str(maxEpi)]);
disp('Total time required execute and plot values ');
disp(dt);         disp('end');
```

ball_Feature_matching.m

```
close all;   clear all;   clc;  disp('hello')
ballImageRGB = imread('C:\Users\rashmi\Desktop\MATLAB-RL-
Fearturebased\feature extraction\foot_ball.png');
[rows, cols, numOfBands] = size(ballImageRGB)
groundImage = imread('C:\Users\rashmi\Desktop\MATLAB-RL-
Fearturebased\feature extraction\soccer ball_ground.png');
groundImageRGB = imresize(groundImage,0.8);
[rows, cols, numOfBands] = size(groundImageRGB)
ballImageGRAY= rgb2gray(ballImageRGB);
[rows, cols, numOfBands] = size(ballImageGRAY)
groundImageGRAY = rgb2gray(groundImageRGB);
[rows, cols, numOfBands] = size(groundImageGRAY)
ballPoints = detectSURFFeatures(ballImageGRAY);
groundPoints = detectSURFFeatures(groundImageGRAY);
figure('Name','Ball in RGB, Gray and 200 Strongest Points');
subplot(1,3, 1); imshow(ballImageRGB);     title('Image of a Ball');
subplot(1,3,2); imshow(ballImageGRAY);   title('Image of a GRAY Ball');
subplot(1,3,3); imshow(ballImageGRAY);
title('200 Strongest Feature Points from Ball Image');          hold on;
plot(selectStrongest(ballPoints, 200));
figure('Name','Ball in Ground in RGB, Gray and 400 Strongest Points');
subplot(3,1,1); imshow(groundImageRGB);
title('Image of a Ground with Ball');
subplot(3,1,2);
```

```
imshow(groundImageGRAY); title('Image of a GRAY Ground with Ball');
subplot(3,1,3);  imshow(groundImageGRAY);
title('400 Strongest Feature Points from netball Image');      hold on;
plot(selectStrongest(groundPoints, 400));
[ballFeatures, ballPoints] = extractFeatures(ballImageGRAY, ballPoints);
[groundFeatures, groundPoints] = extractFeatures(groundImageGRAY,
groundPoints);
ballPairs = matchFeatures(ballFeatures, groundFeatures)
ballPairs_1 = matchFeatures(ballFeatures, groundFeatures,'Method',
'Threshold')
numMatchPoints = int32(size(ballPairs_1,1));
matchedBallPoints = ballPoints(ballPairs(:, 1), :);
matchedGroundPoints = groundPoints(ballPairs(:, 2), :);
matchedBallPoints_1 = ballPoints(ballPairs_1(:, 1), :);
matchedGroundPoints_1 = groundPoints(ballPairs_1(:, 2), :);
figure('Name','Ball matched features with ball in ground');
subplot(2, 2, 1);
showMatchedFeatures(ballImageGRAY, groundImageGRAY,
matchedBallPoints,matchedGroundPoints, 'montage');
title('Putatively Matched Points (Including Outliers) SURF')
[tform, inlierBallPoints, inlierGroundPoints,status] =
estimateGeometricTransform(matchedBallPoints, matchedGroundPoints,
'affine')
subplot(2, 2, 2);
showMatchedFeatures(ballImageGRAY, groundImageGRAY,
inlierBallPoints, inlierGroundPoints, 'montage');
title('Matched Points (Inliers Only)SURF');
subplot(2, 2, 3);
showMatchedFeatures(ballImageGRAY, groundImageGRAY,
matchedBallPoints_1,matchedGroundPoints_1, 'montage');
title('Putatively Matched Points (Including Outliers) SURF');
[tform1, inlierBallPoints1, inlierGroundPoints1,status] =
estimateGeometricTransform(matchedBallPoints_1, matchedGroundPoints_1,
'affine')
subplot(2, 2, 4);
showMatchedFeatures(ballImageGRAY, groundImageGRAY,
inlierBallPoints, inlierGroundPoints, 'montage');
title('Matched Points (Inliers Only)SURF');
ballPolygon = [1, 1; size(ballImageGRAY, 2), 1;size(ballImageGRAY, 2),
size(ballImageGRAY, 1);1, size(ballImageGRAY, 1);1, 1]
newBallPolygon = transformPointsForward(tform, ballPolygon)
newBallPolygon_1 = transformPointsForward(tform1, ballPolygon)
```

```
figure('Name','Ball detected in gray and RGB image');;
subplot(2, 1, 1); imshow(groundImageGRAY); hold on;      axis on;
rectangle('Position', [310,180,150,150],'Edgecolor', 'r');
line(newBallPolygon(:, 1), newBallPolygon(:, 2), 'Color', 'red','LineStyle','--
','Marker' ,'o');
line(newBallPolygon_1(:, 1), newBallPolygon_1(:, 2), 'Color',
'green','LineStyle','--','Marker' ,'o');
title('Detected Ball in Gray');
subplot(2, 1, 2); imshow(groundImageRGB); hold on;        axis on;
rectangle('Position', [310,180,150,150],'Edgecolor', 'r');
line(newBallPolygon(:, 1), newBallPolygon(:, 2), 'Color', 'red','LineStyle','--
','Marker' ,'o');
line(newBallPolygon_1(:, 1), newBallPolygon_1(:, 2), 'Color',
'green','LineStyle','--','Marker' ,'o');
title('Detected Ball in RGB ');
disp('END')
```

APPENDIX E

Mathematical Model of Object Identification

E.1 Detection of Interest Point

E.1.1 Hessian-Based Interest Points

In the image (IMG) point A = (x, y) is considered whose matrix of Hessian H(A, σ) on point A taking scale σ is given by

$$H(A,\sigma) = \begin{bmatrix} L_{xx}(A,\sigma) & L_{xy}(A,\sigma) \\ L_{xy}(A,\sigma) & L_{yy}(A,\sigma) \end{bmatrix} \tag{E.1}$$

Where $L_{xx}(A,\sigma)$ - convolution of 2nd order Gaussian derivative at point A of an image ***I*.**

Similarly, $L_{yy}(A,\sigma)$ and $L_{xy}(A,\sigma)$ are calculated.

For discretization and cropping of an image Gaussians($\sigma=2s$) are required. They are optimal for the analysis of scale space.

$$Img(x, y) = Img(x, y) + Img(x-1, y) + Img(x, y-1) - Img(x-1, y-1) \tag{E.2}$$

E.1.2 Scale Space Representation

Box filters of specific dimensions (9x9, 15x15, 27x27, and so on) are convolved for each scale. Box filters preserve the high-frequency components which get lost in zooming-out of the image. This limits scale invariance. Scale-space is further divided into a set of filter responses, octaves. Each octave is scaled by a factor of 2. For each successive level increase in a minimum of 2 pixels which means one on each side. This keeps the size odd and ensures the central pixel should be present which results in increasing mask size by 6 pixels.

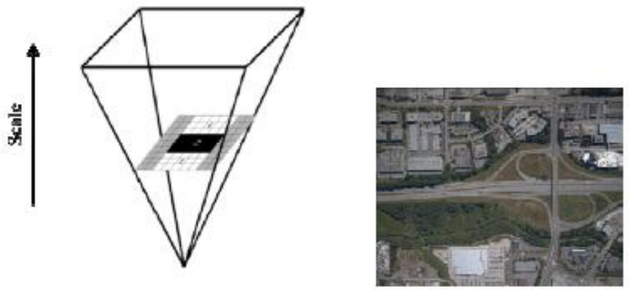


Figure E.1 Scale Space Representation

E.1.3 Localization of Interest Point

Localization of interest points is done by suppression of non-maximum points in the neighborhood of 3x3x3. Interpolation in terms of scale and image space is done for the maxima of the Hessian matrix determinant.

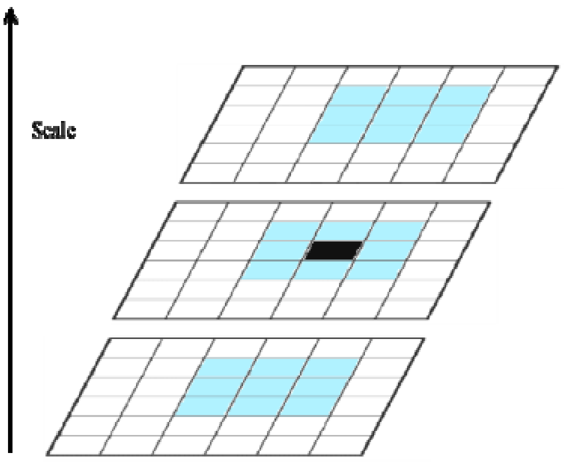


Figure E.2 Interest Point Localization using 3D Non-Maximum Suppression Concept

E.2 Description of Interest Point

E.2.1 Feature Vector

The horizontal and vertical Haar wavelet response is calculated over each subdivision and four metrics are extracted from each subdivision using 5x5 equally spaced points. These metrics are then summed to produce the local feature vector which is concatenated to form a 64-element feature vector that describes the interest point and surrounding neighborhood.

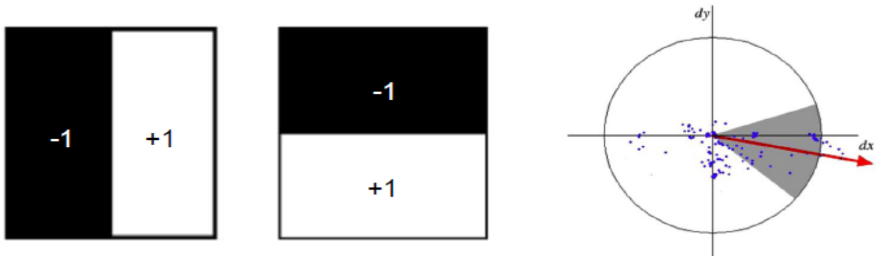


Figure E.3 Haar Wavelet Filters and Sliding Window Orientation

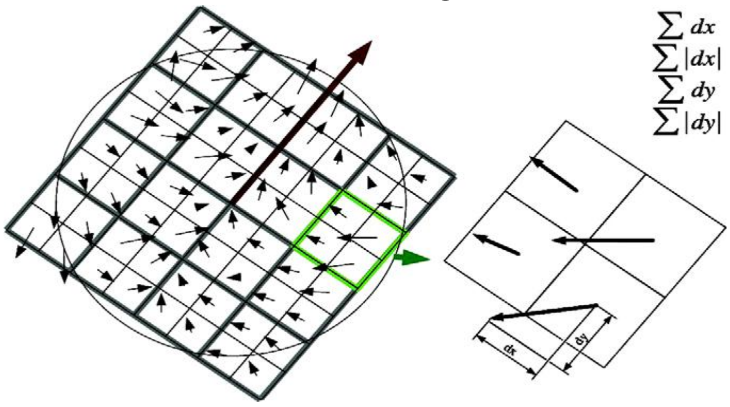
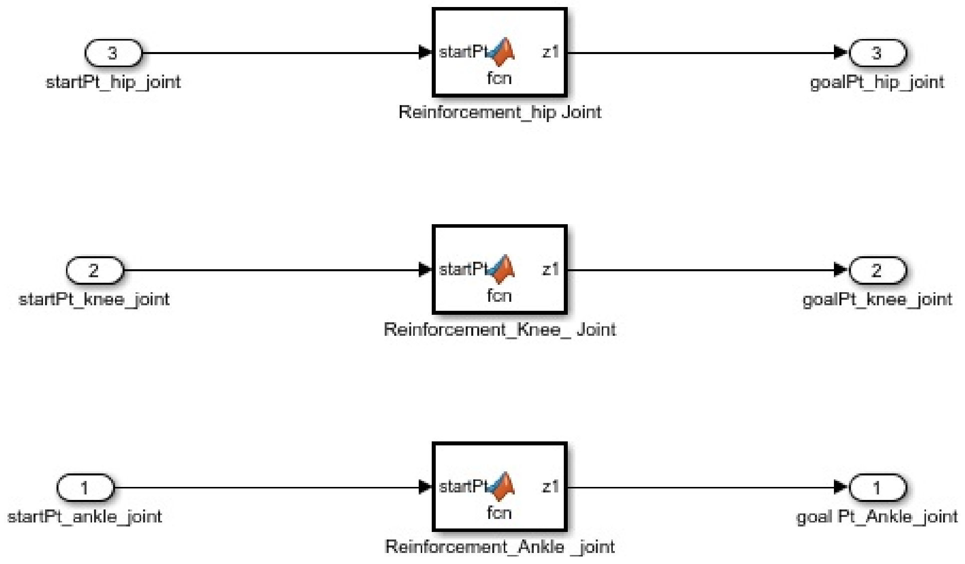


Figure E.4 Descriptor Vector

APPENDIX F

F.1 Simulink Reinforcement Controller



CURRICULUM VITAE

Rashmi Sharma

Research Associate

University of Petroleum & Energy Sciences

Educational Qualifications

- Bachelor of Science (1996) Vikram University Ujjain (M.P.)
- Master in Computer Application (1999) DAVV Indore (M.P.)
- Master in Technology (2010) UPTU, Lucknow (U.P.).

Interest area includes - Soft Computing, Artificial Intelligence, Machine Learning, and Machine Vision.

Papers Published :

1. Rashmi Sharma, Dr. Inder Singh, Dr. Manish Prateek, Dr, Ashutosh Pasricha, (July 2020), " Comparative Study of Learning and Execution of Bipedal by Using Forgetting Mechanism in Reinforcement Learning Algorithm", *Journal Européen des Systèmes Automatisés* (JESA), IIETA(International Information and Engineering Technology Association) Volume 53, Number 3, 2020, pp 335-343 http://www.iieta.org/journals/jesa/paper/10.18280/jesa.530304
2. Rashmi Sharma, Dr. Inder Singh, Dr. Manish Prateek, Dr, Ashutosh Pasricha, (June 2019), "Implementation of Feature-Based Object Identification in Bipedal Walking Robot", International Journal of Engineering And Advanced Technology (IJEAT), Volume-8, Issue-5 ISSN: 2249-8948, pp 110-113.
3. Rashmi Sharma, Dr. Inder Singh, Deepak Bharadwaj, Dr. Manish Prateek,(May 2019), "Incorporating Forgetting Mechanism in Q-Learning Algorithm for Locomotion of Bipedal Walking Robot", International Journal of Innovative Technology and Exploring

Engineering(IJITEE), Volume-8 Issue-7 ISSN: 2278-3075 pp 1782-1787

4. Deepak Bharadwaj, Dr. Manish Prateek, Rashmi Sharma,(May 2019), “Development of Reinforcement Control Algorithm of the lower body of Autonomous Humanoid robot” IJRTE, Vol 8, Issue. 1, pp. 915–919.
5. Rashmi Sharma, Manish Prateek, Ashok K. Sinha, (May 2013)," Use of Reinforcement Learning as a Challenge: A Review" International Journal of Computer Application, New York, (0975-8887) Vol 69-Number 22 pp 28-34 May 2013 DOI:10.5120/12105-8332.

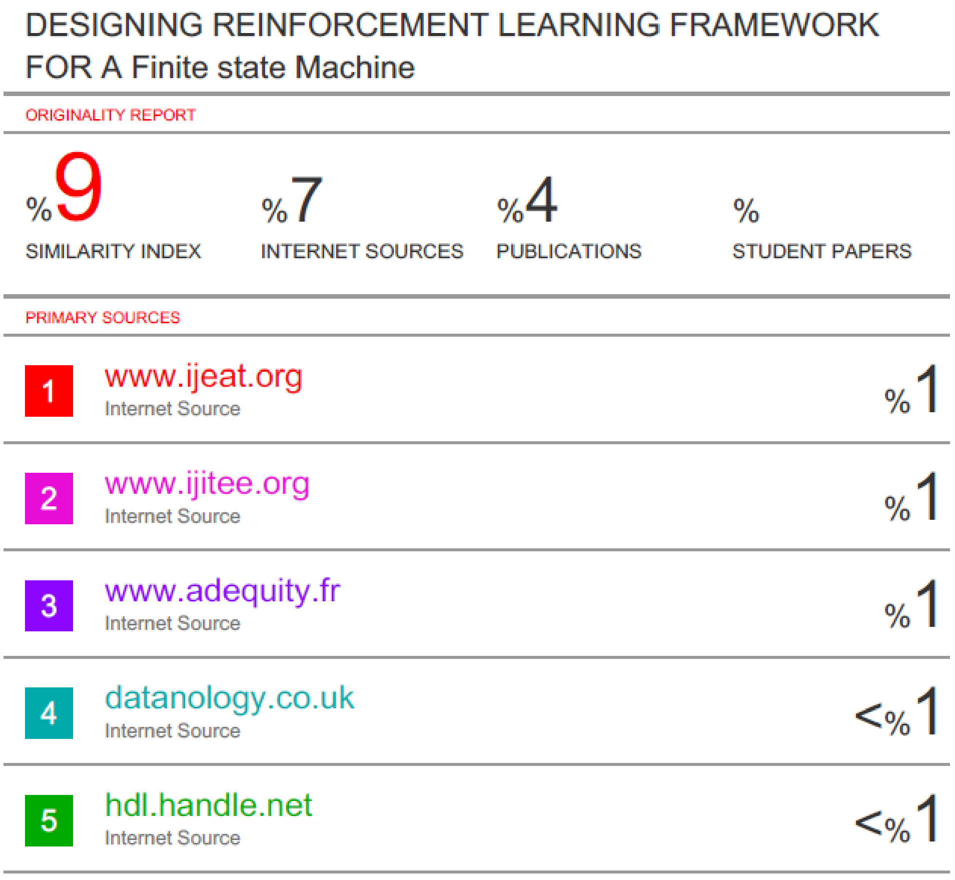
COMMENT INCORPORATED SUMMARY

Below is the summary list of the actions taken upon relevant review comments/ suggestions

Sl.No	Comments/Suggestions	Changes incorporated
1	The proposed methodology would be described with a flow diagram for more clarity.	Changes incorporated on page no 64-66
2	How the proposed methods are comparable with existing image processing techniques	Changes incorporated on page no 90-92
3	Why learning based control techniques as a model-free controller as compared to other conventional schemes?	Changes incorporated on page no 114
4	How the system dynamics and variations of the system will be accounted in the proposed method?	Changes incorporated on page no 67
5	Why only lower body (biped), why not to the complete body (humanoid)?	Changes incorporated on page no 171
6	Why Q-learning and why not other learning (DQN, SARSA, etc.) techniques? (no discussions were there in the thesis)	Changes incorporated on page no 99-102
7	In the scope of future work, it can be mentioned how the same technique could be extend to full- edged system, or some other aspects to modern robotics (for example extending to mobile manipulators, autonomous manipulation, etc along with some reasonable justifications, how it can be extended to these areas).	Changes incorporated on page no 170 and 171
8	Add some references within the last 3 years in the Literature Review of Chapter 2.	Changes incorporated on page no 42-45

9	At the end of Chapter 2.1, add a concluding paragraph to make the article read more smoothly.	Changes incorporated on page no 19
10	Some of the pictures are blurred and the font is too small. Please make them easy to read.	Changes incorporated on page no 51-53, 55-56, 58-59, 61-62, 116-117, 121-122, 124-125, 146, 148, 150-167 in figure . Changes incorporated in tables on page no 127, 130, 138-143
11	This thesis designs a reinforcement learning framework with incorporation of Forgetting Mechanism in Traditional Q-learning Algorithm. However, I cannot find any compared algorithm in the experiment part of Chapter 9. In order to make the thesis more convincing, please add at least one state of the art algorithm about Q-learning into the framework for comparing.	Changes incorporated on page no 90-92,167-168

THESIS PLAGIARISM CHECK REPORT



DESIGNING REINFORCEMENT LEARNING FRAMEWORK FOR A Finite state Machine

ORIGINALITY REPORT

%9	%7	%4	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijeat.org Internet Source	%1
2	www.ijitee.org Internet Source	%1
3	www.adequity.fr Internet Source	%1
4	datanology.co.uk Internet Source	<%1
5	hdl.handle.net Internet Source	<%1