DESIGN, IMPLEMENTATION AND CONTROL OF A HEXAPOD ROBOT USING REINFORCEMENT LEARNING APPROACH

by

Mohammadali Shahriari

A thesis

Presented to Sharif University of Technology, International Campus, Kish Island

in partial fulfillment of the

Thesis requirements for the degree of

Master of Science

in

Mechatronics

Supervisor:

Dr. Amir A. Khayyat

Kish Island,Iran, 2013

Sharif University of Technology
International Campus, Kish Island

This is to certify that the Thesis Prepared:

by: **Mohammadali Shahriari**

Entitled: **DESIGN, IMPLEMENTATION AND CONTROL OF A HEXAPOD ROBOT USING REINFORCEMENT LEARNING APPROACH**

and submitted in partial fulfillment of the requirements for the Degree of

Master of Science

Complies with the regulations of this university and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Supervisor: **Dr. Amir A. Khayyat**

External Examiner: **Dr. Aria Alasti**

Internal Examiner: **Dr. Kambiz G. Osgouie**

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. The work described in this thesis has not been previously submitted for a degree in this or any other university, and unless otherwise referenced it. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

ABSTRACT

DESIGN, IMPLEMENTATION AND CONTROL OF A HEXAPOD ROBOT USING REINFORCEMENT LEARNING APPROACH

Mohammadali Shahriari, M.Sc

Sharif University of Technology, International Campus, Kish Island, 2013

Supervisor: **Dr. Amir A. Khayyat**

Hexapod Robots give us the ability to study walking robots without facing problems such as stability. It has a great deal of flexibility in movement even if some legs become malfunctioned or face some difficulties.

Control of legged robots compared to other kind of locomotions is difficult and requires fairly heavy on-line computations to be done in real time. Therefore, a machine-learning solution may be more convenient in such problems. One method of machine-learning which is used for legged robots with outstanding potential is Reinforcement Learning (RL). RL is a promising approach for achieving complex robots control in dynamic environments. RL techniques are interesting subjects in both cognitive sciences and control theory. In control theory, designing a system that completely works with perfect performance is quite difficult. When unforeseen errors or disturbances are involved within the system it becomes an exhaustive procedure to use classical control theory. By designing a system which has the capability of learning how to accomplish a task on its own, there would not be any need for calculating and prediction

of complicated control algorithms. The ability of learning is a core factor of cognition in cognitive sciences. Free gait analysis of walking robot is one such complex problem. It is to study the learning ability of a six-legged walking robot in a dynamic environment. A simple RL approach is used to develop walking gaits for hexapod. A six- legged (hexapod) robot will be implemented with Reinforcement learning algorithms. The capability of learning how to walk of a robotic hexapod agent is explored using only considering the ability of the robot to move its legs and tell whether it is moving forward. Thus, the hexapod can be seen as an analog for a biological subject which lacks all the instincts excluding the basics which may be seen in infants without external support or parental figure.

In this thesis an 18 degrees of freedom robot entitled as "SiWaReL" is designed and built. The kinematics analysis of the hexapod design is presented using a modular view approach. A dynamic model of the hexapod is modeled using objective oriented programming to execute the developed iterative learning procedure in simulation environment which is used for free gait generation. The results of all formulation and learning control are applied to the prototype, and then verified. This thesis studies learning ability of hexapod walking robot to walk using RL techniques and implements the developed approaches and formulations on a designed experimental six-legged walking robot prototype. The prototype is used for demonstration of the results.

Keywords: *Gait Analysis, Hexapod, Intelligent Control, Reinforcement learning, Robotic*

ACKNOWLEDGMENTS

First and foremost, I offer my sincerest gratitude to my supervisor, Dr Amir A. Khayyat, who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my Master's degree to his encouragement and effort. This achievement has become possible only because of the unconditional support provided by Dr. Khayyat. A person with an amicable and positive disposition, who has always been available to clarify my doubts despite his busy schedules. This is considered as a great opportunity for me to do my Master program under his guidance and to learn from his research expertise. One simply could not wish for a better or friendlier supervisor.

It is with immense gratitude that I acknowledge the support and help of Dr. Amin Mousavi, who as a good friend, gave me this great idea to do as my master thesis. He has always been willing to help, give his best suggestions and guide me through my research for the past several years. He also helped me develop my background in intelligent control, robotics and machine learning. Cheers to Dr. Amin for being a great reliable person to whom I could always talk about my ideas and research lines.

My sincere thanks also goes to Dr. Kambiz G. Osgouie for his great contribution in this thesis and offering me the various opportunities and leading me to working on diverse exciting projects. The thesis would not have come to a successful completion without his assist.

Last but not least, I would like to thank my family; my parents, for giving

birth to me at the first place and supporting me spiritually throughout my life.

Above all, I owe it all to Almighty God for granting me the wisdom, health and strength to undertake this research task and enabling me to complete it.

Contents

Table of Contents viii

List of Figures x

List of tables xiii

1 Introduction and Literature 1
- 1.1 Hexapod Walking Robot . . . 2
- 1.2 Comparison of Wheeled, Tracked and Legged Locomotion . . . 2
  - 1.2.1 Wheeled Locomotion . . . 3
  - 1.2.2 Tracked Locomotion . . . 3
  - 1.2.3 Legged Locomotion . . . 4
- 1.3 Legged Robot Configurations . . . 5
  - 1.3.1 Static Stability and Dynamic Stability . . . 5
  - 1.3.2 Walking Speed of Legged Robots . . . 6
  - 1.3.3 6 Legged Gait Diagrams . . . 7
  - 1.3.4 Existing Hexapod Robots . . . 10
- 1.4 Robot Control . . . 14
  - 1.4.1 Learning Control . . . 16
- 1.5 Summary . . . 18

2 Designing and Building the Prototype of Hexapod Robot "SiWaRel" 19
- 2.1 Actuators . . . 19
- 2.2 Body Parts and Rapid Prototyping . . . 22
- 2.3 Servo-motor Drive Board (Servo Shield) . . . 24
- 2.4 Power Supply . . . 25
- 2.5 Low Level control of the Robot . . . 25
- 2.6 Summary . . . 26

3 Modular View Kinematic Analysis of Hexapod 29
- 3.1 Introduction . . . 29
- 3.2 Kinematic Analysis . . . 30
  - 3.2.1 Inverse Kinematic of Hexagonal Hexapod Robot . . . 30
  - 3.2.2 Forward and Inverse Kinematic Analysis of one Leg . . . 34

3.3 Gait Analysis of Hexapod Robot 38
3.3.1 Simulations of Gait Analysis 40
3.3.2 Implementation Gait Analysis and Inverse Kinematic Formulations on SiWaReL Prototype 43
3.4 Summary 46

4 Dynamic Modeling of SiWaReL Hexapod Robot 47
4.1 Hexapod Robot Model in Dynamic Simulation Environments 48
4.1.1 V-rep, Coppelia Robotics 48
4.1.2 Webots, Cyberbotics 49
4.2 Dynamic Modeling of Hexapod Robot in MATLAB© SimMechanics . 50
4.2.1 Leg Design 51
4.2.2 Hexapod Design 51
4.3 Summary 55

5 Reinforcement Learning in Hexapod Walking 56
5.1 Reinforcement Learning Problem Architecture 57
5.2 State and Actions Specification in SiWaReL for Walking 58
5.3 Fuzzy Reward 63
5.4 Action Selection 67
5.5 Learning algorithm 69
5.6 Results 73
5.6.1 Random Action Selection 73
5.6.2 ϵ-greedy Action Selection 76
5.6.3 Decision making 79
5.7 Summary 79

6 Conclusion 82

References 83

Index 86

List of Figures

1.1 A multi-legged robot's Support pattern and the Center of Mass (CoM) within the support polygon. Static balancing occours when the CoM of the Robot lies within the support pattern. 5
1.2 Two typical types of hexapod robots, hexagonal (left) and rectangular (right) 7
1.3 Beetle's leg structure 8
1.4 4 gaits inspired by insects leg movements. Swing phases are shown by black bars and the empty space between these bars represents the support phase. Gait 1 is static stable tripod gait, the other gaits are dynamic natural gaits observed from insects such as cockroaches. Gaits 2 to 4 due to the fact that only 2 legs are in stance phase in a time are not statically stable. In Gaits 2 to 4 only the sequences of the legs are different which make different gaits. The Legs labels are shown at the top besides the insect diagram 9
1.5 Another type of wave gait which 2 legs are in transfer phase while the other 4 are in support phase 10
1.6 Tripod gait. Since three legs move at the same time, it keeps static stability 10
1.7 Rhex hexapod robot, legs of the robot by swinging circular motions make the robot move. RHex moves fast and robust in forward direction by speeds up to one body length per second (0.5 m/s) [1] 11
1.8 Webx hexapod design. This project is found on the Internet. This robot uses Atmel AT90S8515 processor at 8 Mhz and some RC components. This is a hobby project 12
1.9 Some designs of hexapod robot built with RC-servo motors. 12
1.10 California University of Technology, Hopping hexapod robot prototype which can jump. 13
1.11 Timberjack, a walking machine project by Pulstech. The control and walking procedure is done by an intelligent computer. 15
1.12 Robot Control block diagram architecture 17

2.1 Designed 3D model of 18 DoF hexapod 20
2.2 3D model of leg and servo-motors. 21

2.3 20× (18+2 spares) GWS 03T/STD/J Servomotors are provided for making SiWaRel Prototype. 21
2.4 Making body parts for SiWaReL hexapod 22
2.5 Assembled SiWaRel hexapod walking robot prototype 23
2.6 Shanjing Power Supply, input:100-240 Volts and 1.2 A 50/60Hz, output:5V 7A 24
2.7 The diagram shows how to implement the real-time connection between the robot and MATLAB. 26
2.8 Serial Control board and servo shield 27
2.9 SiWaReL prototype with real-time connection to MATLAB. 27

3.1 18 DoF SKPRbot six-legged robot design 31
3.2 The hexapod leg design inspired by a spider leg 31
3.3 Hexagonal hexapod coordinate frame assignment, ground frame O and trunk frame O' 32
3.4 Link-frame assignments for finding Denavit Hartenberg parameters . 35
3.5 A 3 DoF hexapod leg design link assignment and parameters for inverse kinematic analysis of one leg 37
3.6 18 DOF SKPRbot six-legged robot design 39
3.7 Tripod Gait and Wave Gait signals sequences 39
3.8 Joint space variables of the hexapod robot in tripod gait through one complete step (left), Tripod walking gait simulated leg sequences in one complete step. Triangles are grounded leg tips (right). 41
3.9 Joint space variables of the hexapod robot in wave gait through 1 step (Left), Wave walking gait simulated leg sequences in one complete step. (Right). 42
3.10 Tripod gait implementation on SiWaReL prototype in 2 steps 44
3.11 Wave gait implementation on SiWaReL prototype in one step 45

4.1 V-rep robot simulation environment. A hexapod walking robot is simulated here from its own library and walking algorithms. 48
4.2 Webots robot simulation environment 50
4.3 SimMechanics Model a biologically inspired 3 degrees of freedom Leg 51
4.4 Model of 3 DOF leg 52
4.5 Ground contact model in SimMechanics 53
4.6 Complete model of 3 DOF Leg with ground contact model 54
4.7 Hexapod robot design in SimMechanics 54
4.8 Dynamic modeling of six-legged walking robot in MATLAB Simulink, SimMechanics 55

5.1 The diagram of reinforcement learning procedure. Actions taken by agent lead to certain states and reward. The goal is to find the best policy to take actions with the highest rewards. 57
5.2 The defined states for reinforcement learning of walking in hexapod . 59

5.3 State action space with omitted non-sense actions regarding walking cycles 61
5.4 The diagram of fuzzy system which is used to generate reward values using sensors data (with respect to actions and states) 63
5.5 The surface of fuzzy system with respect to different inputs. Due to inputs characteristics similarity excluding Δ_x which is shown in (b) for example, the other surfaces will be the same as these two surfaces. . . 64
5.6 Signals of both functional reward which is mathematical formula and fuzzy reward. As shown, fuzzy reward signal is more detailed and accurate. 65
5.7 The diagram of Λ fuzzy reward 65
5.8 The surface of Λ fuzzy reward with respect to Δ_x & Λ. 66
5.9 Comparison of reward evaluation of Λ fuzzy reward, the normal fuzzy reward and functional reward. It is shown that Λ fuzzy reward is slightly less accurate than the normal fuzzy reward. 67
5.10 Policy search for hexapod walking learning 69
5.11 The schematic of reinforcement learning for hexapod robot 71
5.12 The progress of SiWaRel learning to walk iterations 74
5.13 The progress of SiWaRel learning to walk iterations 75
5.14 Updated Q matrix in 10000 learning iterations 76
5.15 Updating trend of Q matrix to 150000 learning iterations with ϵ-greedy action selection, $\epsilon = 0.1, 0.2$ & 0.5 77
5.16 Updated Q after 150000 Iterations with different greediness. The right figures show the explored state action space. 78
5.17 Evaluating the optimal policy on the dynamic model of hexapod robot. As it can be seen, the robot is moving on the desired direction $(+x)$. 80

List of Tables

3.1 Denavit-Hartenberg parameters of 3 DoF hexapod's leg 38

5.1 The joint variables of six-legged walking robot in different defined states 62
5.2 Speed performance comparison of different Rewarding systems 66

Chapter 1

Introduction and Literature

This thesis is about the study and development of learning ability of walking robots to walk using Reinforcement Learning (a machine learning approach) techniques and implementing the developed approach on a specific experimental six-legged walking robot. A real hexapod robot is designed and built based on Google SKPRbot design for demonstration of the developed approach.

During this study, three concepts have been considered to work on. The first one is the kinematic analysis of the Hexapod walking robot. The inverse kinematic analysis is needed for formulating that how the robot should move in joint space to walk on a desired trajectory. The second one is the dynamic model for the robot for simulation of learning procedure. The third and the final concept of this thesis is to study the architecture and implement Reinforcement learning algorithms on the dynamic model to make the robot learn walking.

1.1 Hexapod Walking Robot

A hexapod mobile robot is a mechanical vehiecle that has the ability to walk. As a robot could be stable statically standing on three or more legs, a six legged walking robot can be highly flexible in movements and perform different missions without dealing with serious kinematic and dynamic problems unlike quadrapods and other kind of legged robots. In hexapods even if one or two legs become malfunctioned or fail to work, the robot still can have the ability to walk and continue its mission. Furthermore, all legs of the robot are not needed to make the robot stable and this gives the robot the capability to use the other legs for reaching new foot placements or manipulating a payload [2].

Due to movement flexibility and stability of hexapod robots, this kind of walking robot is chosen for the study of walking learning using Reinforcement Learning (RL).

1.2 Comparison of Wheeled, Tracked and Legged Locomotion

Although in the nature, wheeled locomotion has no place, the early developed vehicles had wheels for movement due to invention of steam engines, railways and combustion engines. Transportation was so easy using wheeled technologies. However when it came to rough surface and unknown terrains, wheeled vehicles were not proper. This challenge led to the development of the tracked (palette) locomotion in order to overcome this problem. There are also some drawbacks of the tracked locomotion such as destroying the terrain as the vehicle passes through. Legged locomotion is an alternative locomotion to both tracked and wheeled locomotion by imitating walking animals in the nature [3].

The most important elements which determine the locomotion and robot properties are the terrain characterizations. Geometric, material and temporal properties such as roughness and inclination, friction and hardness, different surfaces (e.g.under or over water) respectively are three properties that Hardarson has cited [4]. It is necessary to identify the terrain properties, mission and the criteria and then choose the robot characteristics design. The advantages and disadvantages of these three kinds of locomotion are discussed here.

1.2.1 Wheeled Locomotion

The most common and advantageous locomotion is wheeled which can perform smoothly and fast on even and hard terrains. It can carry high payload and energy efficiency is also a considerable advantage of its kind. The control and implementation of wheeled locomotion is simple compared with the other types as a technical view point. The simplicity makes these systems the most common and attractive ones. However when it comes to uneven, smooth surfaces, obstacles or holes bigger than the radios of the wheels, this kind of locomotion cannot perform well or it may be impossible. For example in sandy or muddy surfaces, dragging the vehicles by the wheels are not possible.

1.2.2 Tracked Locomotion

In unknown terrains, an advantageous locomotion is track locomotion, especially on a soft or loose surface. The palettes contact the ground on a wide surface which provides enough drag for the vehicle to move on the different surfaces such as snowy or sandy ones. It has also a simple control design and can carry high payloads; however, high energy consumption is an important drawback of tracked systems. High weight and

energy loss due to friction lead to making this locomotion not energy efficient. The other disadvantage is destroying the terrain where the tracked vehicle moves on. The vehicle moves on the ground with inflexible palettes that use slip friction; therefore the ground surface would highly be damaged.

1.2.3 Legged Locomotion

Legged systems are the appropriate solution for uneven, rough or loose surfaces. Legged systems do not contact the surface in a continues matter unlike tracked and wheeled locomotion. Due to isolated footholds, legged robots have the capability to choose proper places to step on and move. Isolated footholds make the legged robots have high performance while continuous ground contact in tracked and wheeled systems limits their performance by having undesirable parts of the surface. Legged robot can manage how to distribute force among legs and ground interaction which consequently gives the robot active suspension. Despite the ground characteristics, the legged robot can move the body in any orientation that is desired. The disadvantages of legged locomotion are energy consumption, complex kinematics, dynamic mechanisms and difficult control algorithms. Due to active suspension of legged systems which should support legs in continuous matter, the payload of these systems is considerably lower than other two kinds of locomotion. In legged locomotion, a motor which is the heaviest part usually drives each joint, so this makes the legs heavier than the main body and results in low payload capacity [5]. Different designs of motor placements in leg links in multi-legged robots exist which some are discussed by Gonzales de Santos, Lin and Song, and Clark [6–8]. There are applications for multi-legged robot due to their advantages compared with wheeled and tracked vehicles that can be mentioned such as particle gathering from unknown surfaces (territorial or extraterritorial surfaces), recovery missions in disasters and hazardous areas like

an earthquake and fire, a forest for harvesting trees [4] and de-mining [6],

1.3 Legged Robot Configurations

There are different and enormous configurations of legged robots, from Biped (two-legged) like humanoid robots, Quadrupeds (4-legged), Hexapods (six-legged) and Octopod (eight-legged) robot configurations like spiders. The Hexapods' walking gaits are more statically stable than Quadruped due to their wider combinations of legs.

1.3.1 Static Stability and Dynamic Stability

Stability is one of the fundamental performance characteristics of locomotion analysis. In short words, stability is robot's balancing. In general, two types of stability have been defined: Static stability and Dynamic Stability. Static stability is when the robot is balanced without any force or additional movement. The legs' ground contacts

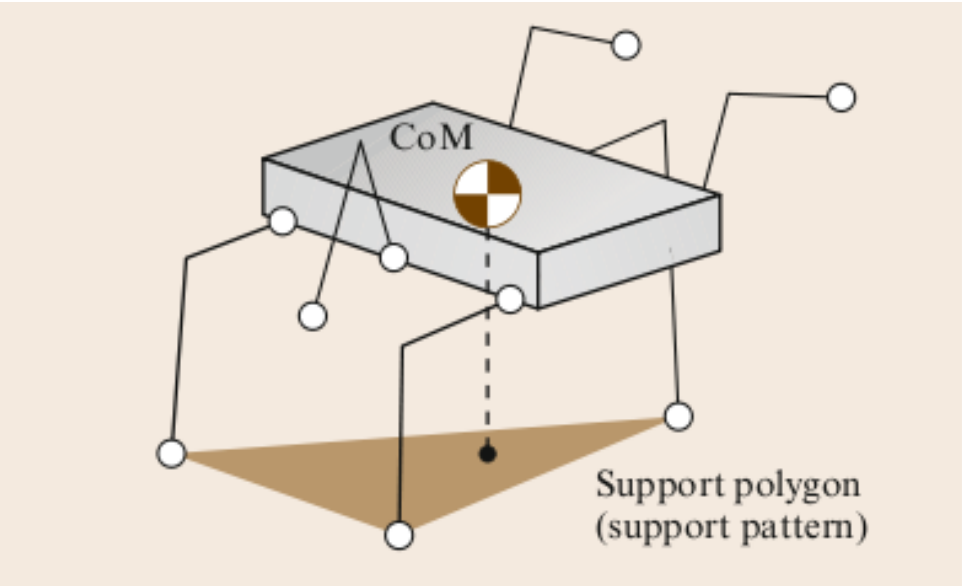


Figure 1.1: A multi-legged robot's Support pattern and the Center of Mass (CoM) within the support polygon. Static balancing occours when the CoM of the Robot lies within the support pattern.

make support pattern of legs and while the Center of Mass (CoM) of the robots lies down within the support pattern, it is statically stable, see figure 1.1 [9].

Dynamic Stability is required at times that the CoM is not above the support pattern. In such cases the robot will fall when no extra movement or forces applied to balance. [10, 11]

A six-legged walking robot has a greater deal of static stability than 4-legged robot due to increasing the support pattern because of more legs. Static walking is possible by moving one leg while the other five legs are in supporting pattern unlike quadrupeds which only minimum of three legs are in supporting of static stability. So Hexapods are more stable than Quadrupeds due to bigger support polygons [12].

1.3.2 Walking Speed of Legged Robots

One of the other robot locomotion key factors is speed. It is shown that V_n the multi-legged speed can be defined as equation below where n denotes number of robot's legs, in wave gait, is related to step size L_s, walking cycle time T, and the duty factor β. β has a direct relation to n [13]. Transfer phase t_T, the time that a leg is on the air and is placing into new position, is $t_T = (1 - \beta)T$ and stance phase t_S, the time in which leg is on the ground, is $t_S = \beta T$ are derived form T and β.

$$V_n = \frac{L_s}{T\beta} = \frac{L_s}{t_T}\left(\frac{1-\beta}{\beta}\right) \tag{1.1}$$

N-legged robot has a minimum duty factor of $\beta_n = 3/n$ [13]. Hence the speed of Quadrupeds, Hexapods and Octopods are determined as $V_4 = 0.33(L_s/T)$, $V_6 = (L_s/T)$ and $V_8 = 1.6(L_s/T)$ respectively. So as we can see, when the number of legs increases, the robot can walk faster in wave gait. It should be mentioned here that these values are derived in wave gait which is a static one.

Gait analysis is the study of walking mechanism and leg sequences which leads to

walking. Different gaits have been developed and studied for many years in different terrains and conditions. [12]

1.3.3 6 Legged Gait Diagrams

There are lots of leg configuration designs studied for hexapod robots. In a general view, hexapods can be categorized into two designs, rectangular and hexagonal, see figure 1.2. Rectangular hexapods are inspired from insects, those having six legs divided in two sides of their bodies. Hexagonal hexapods have six legs distributed axisymmetrically around the body (circular). Each leg of the robot can have two to six degree of freedom which is inspired form the insects legs. Figure 1.3 shows the structure of insects leg which is the base of inspiration for robotic leg design.

Rectangular shaped design are more natural looking and based on animals. This robots are fast in moving straight forward but are not flexible in turning sideways or turn around. Besides, despite the fact that hexagonal designs do not have natural gaits, they are efficient in moving in all directions due to symmetric design. Although rectangular gaits are applicable to hexagonal design, they vice versa, are not always feasible or possible. In hexagonal design, each leg has the same step size in all

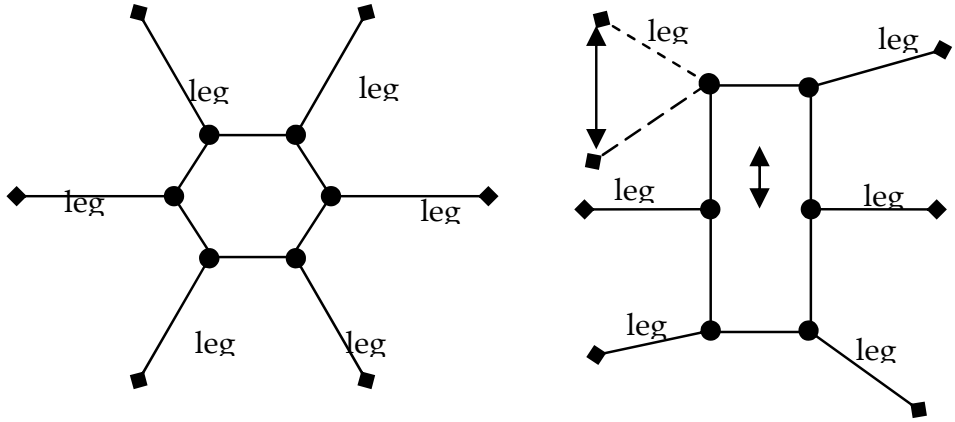


Figure 1.2: Two typical types of hexapod robots, hexagonal (left) and rectangular (right)

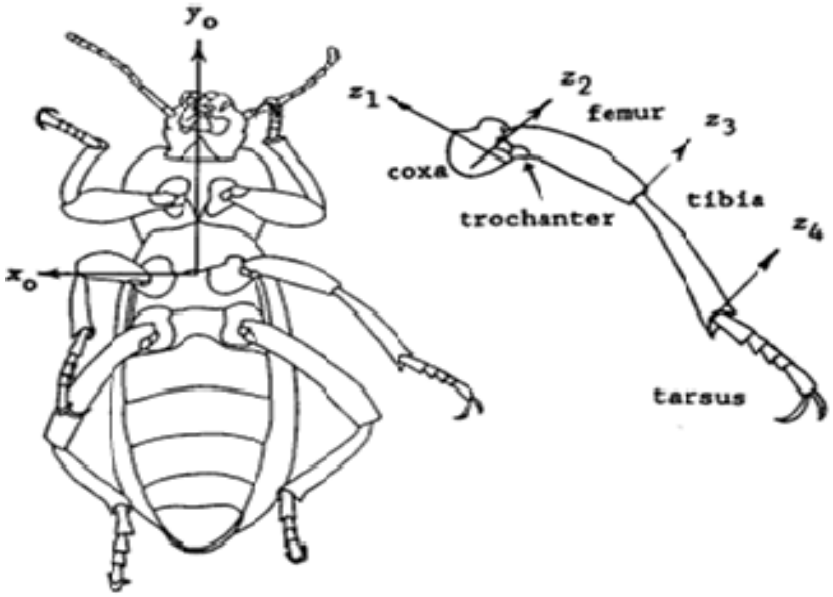


Figure 1.3: Beetle's leg structure

directions while rectangular robots are designed to have large steps in forward or backward direction not in sideways. The static stab of both designs are the same in a general view. Insects have different leg movement sequences, locomotion speed and patterns. In all of these typical gaits, static stability is satisfied which means supporting legs, those make the supporting polygon, always have the CoM inside. The leg movement in gait analysis as mentioned before can be studied in two phases: the support phase and the transfer phase, which are the phases that the leg is on the ground and swinging to a new step on the ground respectively. Considering that the most important element in gait analysis is swing phase and its timing, different gaits can be defined by specifying the swing timings and sequences. Some biologically inspired gaits are shown in figure 1.4

Insects use different gaits such as wave gait, in which one leg moves at a time and is slow to tripod gait, in which three legs move at a time and is fast [14].

There is another type of wave gait in which two legs move as the other 4 legs are

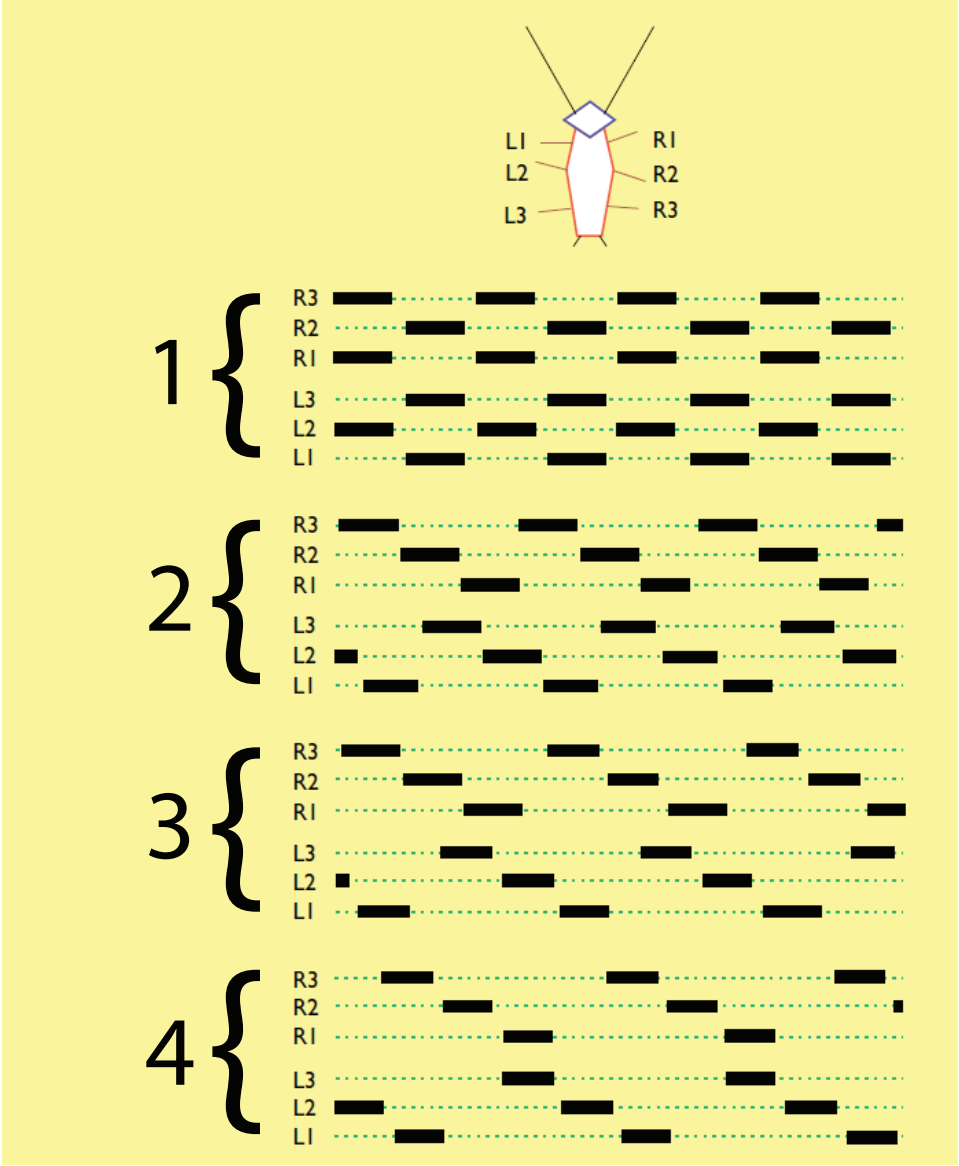


Figure 1.4: 4 gaits inspired by insects leg movements. Swing phases are shown by black bars and the empty space between these bars represents the support phase. Gait 1 is static stable tripod gait, the other gaits are dynamic natural gaits observed from insects such as cockroaches. Gaits 2 to 4 due to the fact that only 2 legs are in stance phase in a time are not statically stable. In Gaits 2 to 4 only the sequences of the legs are different which make different gaits. The Legs labels are shown at the top besides the insect diagram

left in stance position. This gait moves faster than normal wave gait that only one leg moves [9].

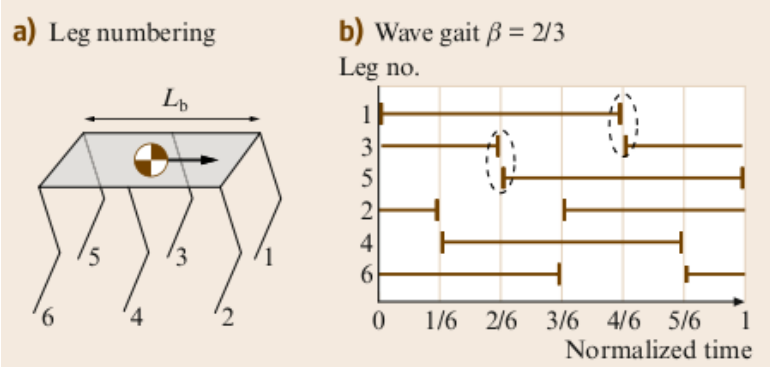


Figure 1.5: Another type of wave gait which 2 legs are in transfer phase while the other 4 are in support phase

Another example of wave gait is shown in figure 1.6, moving as 3 legs in transfer position. This leads to $\beta = 1/2$ and regarding discussed constraints, it is the smallest duty factor that a hexapod could have. Therefore, it is the fastest gait and still statically stable. Due to that, the supporting pattern forms a triangle while this moving gait is called tripod gait.

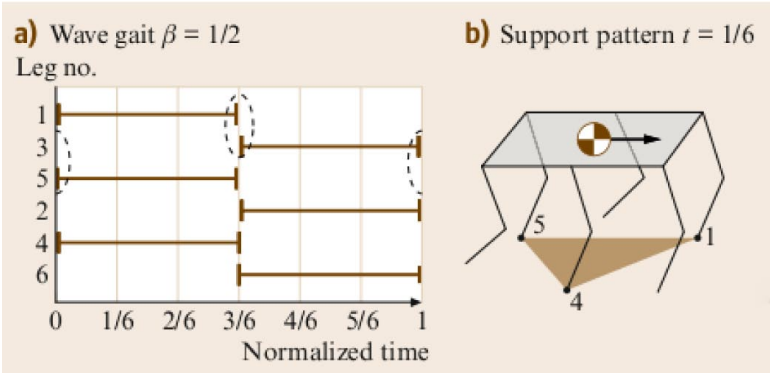


Figure 1.6: Tripod gait. Since three legs move at the same time, it keeps static stability

1.3.4 Existing Hexapod Robots

There are so many different designs of hexapod robots. But the most referred robot on the Internet and journals is the Rhex design which has a very simple and effective

structure which provide the capability to walk over hard terrains with high speed [1, 15]. Some designs for different purposes are shown in figure 1.7. The main problem

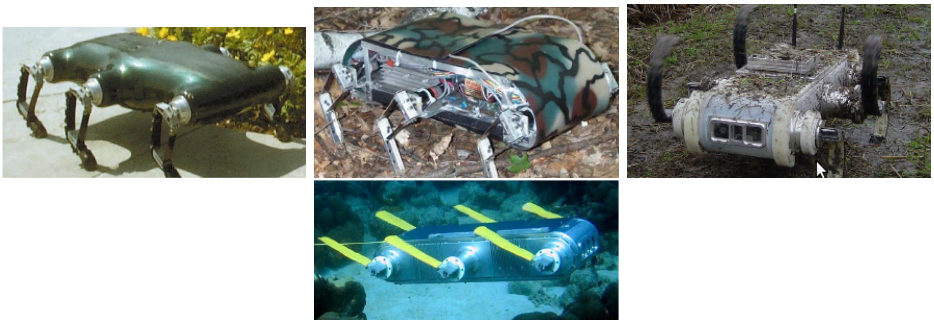


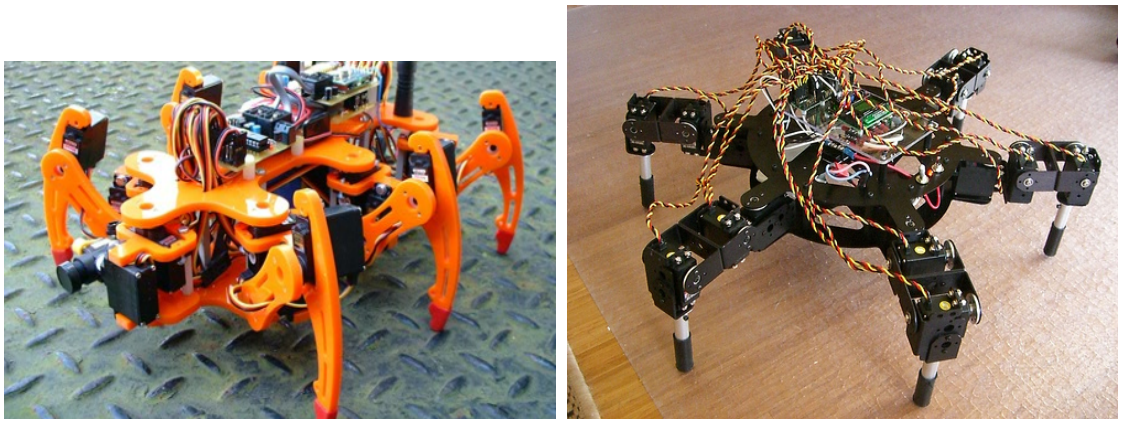
Figure 1.7: Rhex hexapod robot, legs of the robot by swinging circular motions make the robot move. RHex moves fast and robust in forward direction by speeds up to one body length per second (0.5 m/s) [1]

with the Rhex robot design is its lack of capability to turn around, although it can move easily and fast forward or backward in straight line. In a project this problem is shown by building a full scale model [16].

Another example of hexapod design is webx using only RC-servo motors and cheap materials [17]. The leg and body of this robot is shown in figure 1.8.

Eggshell hexapod robot is a design similar to webx with RC-servo motors. There is different circular design by Hexateuthis that all the legs are symmetric around the body and as discussed before has more flexibility in movement than other designs. Figure 1.9 shows these two designs. There is an interesting site for different designs and demonstration of robots http://www.robots-dreams.com/. Another interesting design is a hopping hexapod made by California University of Technology which can jump for missions on extraterritorial terrains which the gravity is small like moon surface. Integrated springs are used to store energy from hopping and landing and a motorized spool compresses them. As tests were performed it is shown that this robot has the capability of jumping vertically 35 cm and angled distance of 50 cm with 30°. The Caltech hopping robot design is shown in figure 1.10 [18].

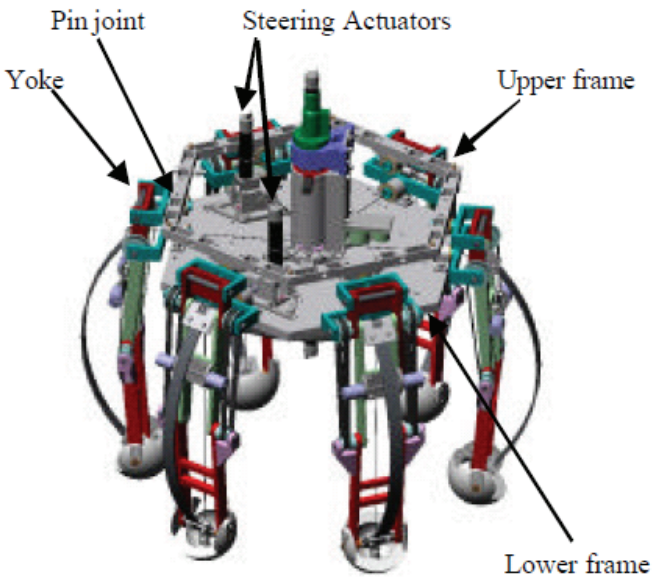
Figure 1.8: Webx hexapod design. This project is found on the Internet. This robot uses Atmel AT90S8515 processor at 8 Mhz and some RC components. This is a hobby project



(a) The eggshell hexapod.

(b) The circular Hexateuthis robot.

Figure 1.9: Some designs of hexapod robot built with RC-servo motors.



(a) Basic overview of the jumping robot.

(b) Leaning left. (c) Standing in neutral position. (d) Leaning right.

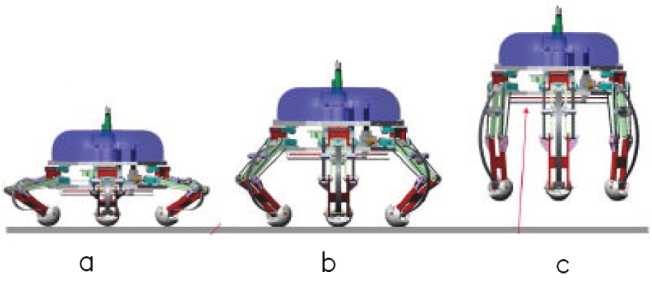


Figure 1.10: California University of Technology, Hopping hexapod robot prototype which can jump.

Pustech has developed an impressive 6 legged robot. The goal of this product was having the best stability with minimum terrain impact. This is an industrial robot for working in hard terrains such as forest while carrying high payloads. It can work with great maneuverability and its walking structure like height of the body or steps can be adjusted based on the ground surface clearance and obstacles height 1.11 [19].

1.4 Robot Control

As literature shows, there is a growing interest in walking robots area, especially in mechanism and control of gait generation. Talking more specifically, statically stable gaits are one of the subjects in robot control that has got more consideration in legged robot control. The main idea is to generate the leg and body movement sequences which leads to the robot's moving in desired direction to complete the given task. The control problem of walking robot, therefore, can be defined as the trajectory planning of the legs. Therefore, the control problem can be defined as finding statically stable gaits and leg trajectory planning.

There are various approaches for controller implementation of different gaits. The definition of a gait is a leg motions coordinate sequence with motions of body sequence for the overall desired body movement. When the gait is to repeat a certain sequence, it is called periodic gait. Hence, the gaits can be classified to periodic gaits, free gaits (non-periodic gait) and combination of both gaits. Periodic gaits work greatly for smooth terrains. Some work can be found in researches by Song, Zang and Waldron [20, 21]. When it comes to rough surfaces with unpredictable obstacles, periodic gaits are not suitable choices. Various analytical and graphical approaches on free gaits are done in developing free gaits. The challenging part of this kind of gaits is the interaction and complexity. Genetic algorithms (GAs) are used to imple-

Figure 1.11: Timberjack, a walking machine project by Pulstech. The control and walking procedure is done by an intelligent computer.

ment some of these approaches. GAs are so useful in numerous learning approaches and have shown that they are efficient in global optimization.

1.4.1 Learning Control

Learning techniques provide the capability of adapting the structures and control laws for the controllers to improve their performance with respect to time, i.e., as they work. This kind of controller, which can be said as adaptive controllers, can be just the same classical controllers with adjusted parameters which adapt or update to reach improved control performance.

In this project, reinforcement learning technique is used to find free gait by just adjusting some specific leg movement and states. This control architecture which is an online learning technique finds the best gait based on what it learns on specific terrain when the robot is currently walking on. Based on what it sees from the environment and explores the terrain by taking some actions, it finds out the best way to walk with possible highest performance.

For research on gait control and other analysis, a semi-autonomous hexapod robot with radially symmetric architecture entitled as (Six-Legged Walking Robot implemented with Reinforcement learning) "SiWaReL" has been developed. The architecture of control is divided into two different layers, low level control and main control. The low level control is about to manage how to send signals to actuators, manage the time sequences and generate suitable pulse with modulation to motors. In the main control all the computation of inverse kinematic analysis and gait analysis is done and fed the found data to the low level control to apply it on the robot.

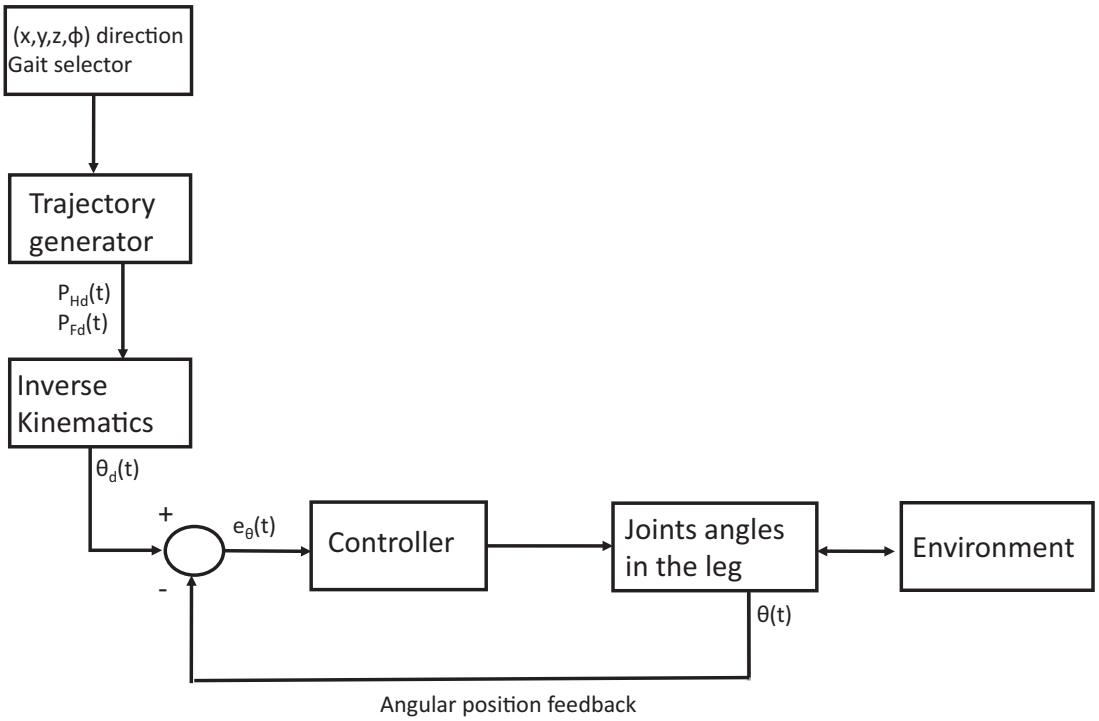


Figure 1.12: Robot Control block diagram architecture

1.5 Summary

In this chapter the hexapod (six-legged) walking robot has been defined completely considering different aspects. Some existing hexapods and their designs are introduced. Walking robots and other kind of locomotion are discussed here, benefits and drawbacks of each kind of locomotion are explained. Different concepts of walking robots like stability, gait analysis and control are studied.

This project is to study six-legged walking robot with a radially symmetric design. The design, kinematics formulation, dynamic modeling and learning control are discussed in following chapters.

Chapter 2

Designing and Building the Prototype of Hexapod Robot "SiWaRel"

In order to perform real demonstration of hexapod control with reinforcement learning approaches and verification of kinematic analysis, a real prototype of hexapod robot entitled as **Si**x-legged **Wa**lking **R**obot implemented with **Re**inforcement **L**earning "SiWaReL" is built. The capability of real time connection with a computer is required for the prototype for online control.

2.1 Actuators

18 servo-motors are required to actuate 18 DoF hexapod robot. 3 servo-motor for actuating every leg based on design 2.2. Each revolute joint is implemented with a servo-motor. Two joints (tibia and femur) are moving in a surface angled with vertical axis by another revolute joint (coxa).

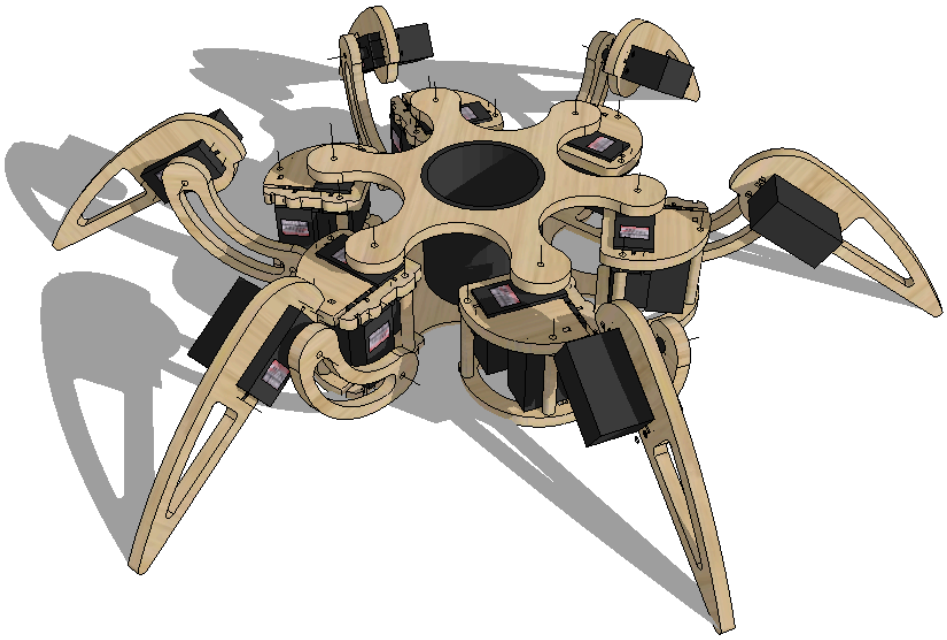


Figure 2.1: Designed 3D model of 18 DoF hexapod

All servo-motors are chosen the same as seen in figure 2.3, **GWS 03T/STD/J**. These kinds of servo-motors can provide 8 kg/cm torque with 6 volts input and from the speed point of view, they can move 60 degrees in 0.27 seconds.

A servo-motor consists of a DC motor coupled with a position feedback and a gear box. This mechanism provides precise angular position control. Talking more specifically, a servomotor is a servomechanism which controls its motion and final angular position by its own internal closed loop control. These kinds of servos which are used here are 3 wire type. The red, brown yellow wires connect the servo to the source (+Vcc), ground (0 GND) and servo input pulse respectively. The yellow wire gets the input signal which indicates the angular position that the motor should go. In other words, the signal indicates the internal control loop position set point.

Some electronic parts like resistors, capacitors, pin headers and wires are also needed for the pull ups or providing sufficient current for driving the servo-motors.

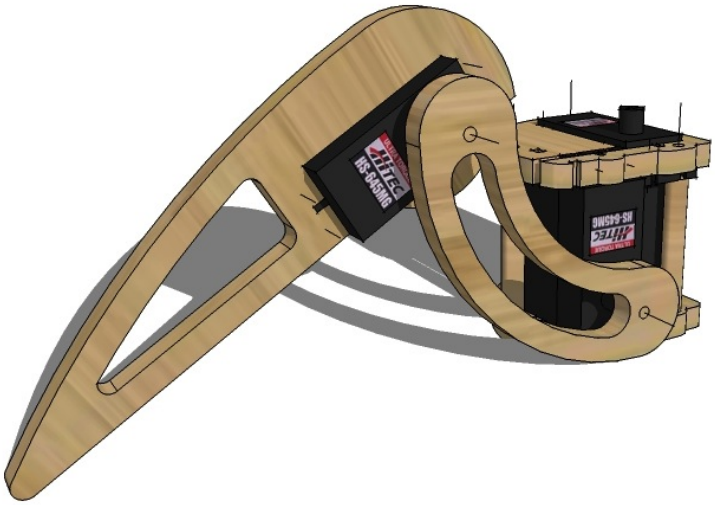


Figure 2.2: 3D model of leg and servo-motors.

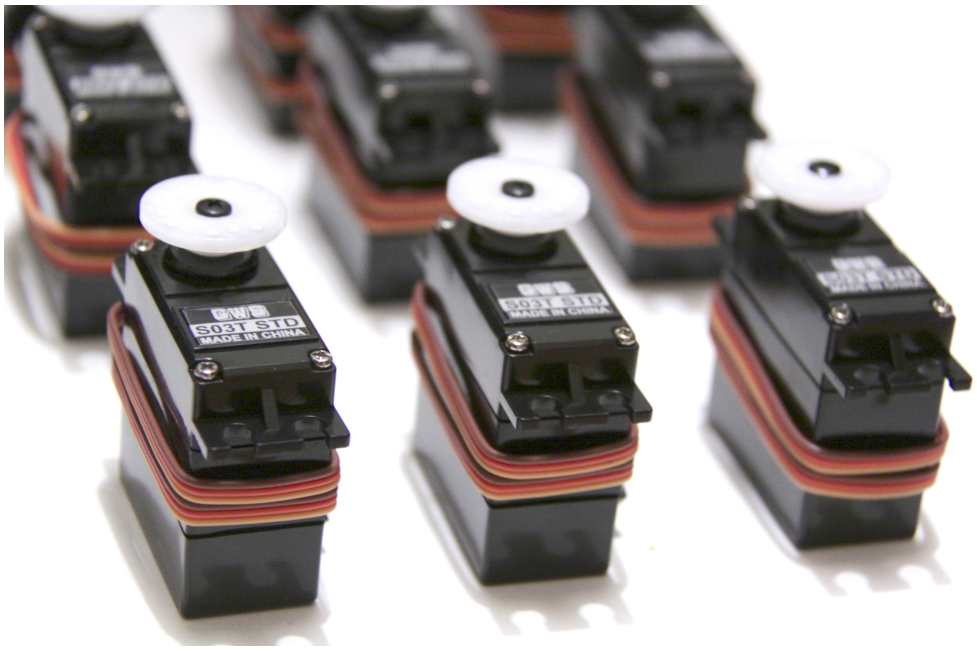
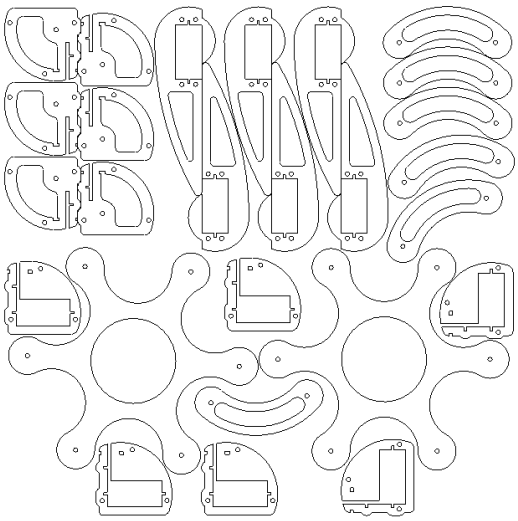
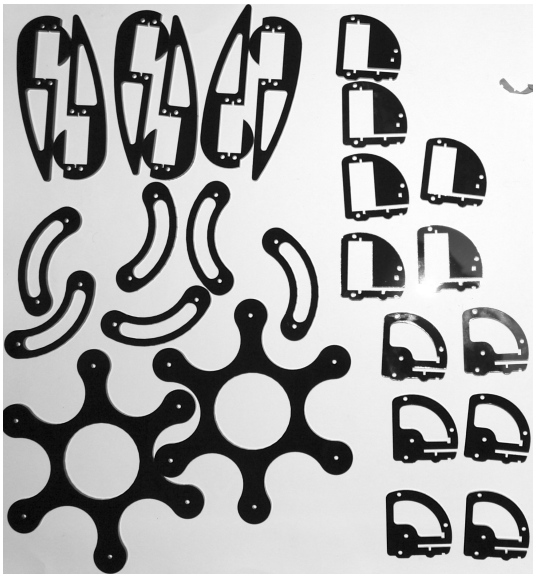


Figure 2.3: 20× (18+2 spares) GWS 03T/STD/J Servomotors are provided for making SiWaRel Prototype.



(a) CAD files of body parts for laser cutting machine



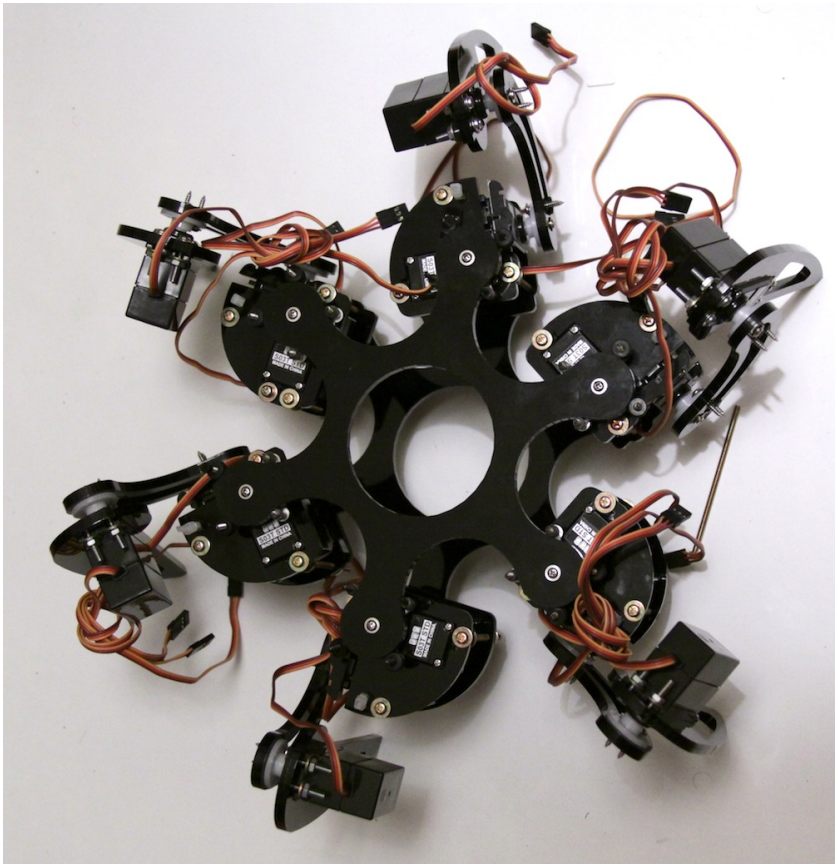
(b) Cut body parts out of plexiglass using laser cutting machine

Figure 2.4: Making body parts for SiWaReL hexapod

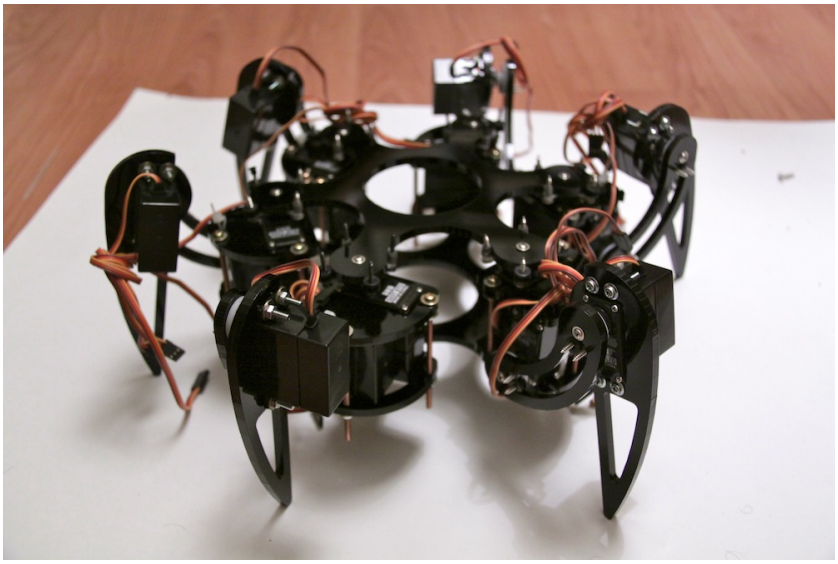
2.2 Body Parts and Rapid Prototyping

To get the body parts, the best way is to use rapid prototyping techniques such as 3D printers and laser cuttings. The material which has been chosen for the body parts is Plexiglass which is a trademark of Poly(methyl methacrylate) thermoplastic. This material is easy to cut using laser cutting machine and is light weighted, but compared to other polymers its impact strength is a bit lower. However, as seen in various projects, this kind of material is good for making body links and parts of experimental prototypes.

The CAD model of hexapod body parts is designed as it is shown in figure 2.4. The body parts are cut out of plexiglass with thickness of 6mm using the laser cutting machine.



(a) Upper view of SiWaReL prototype



(b) Built prototype of SiWaReL hexapod

Figure 2.5: Assembled SiWaRel hexapod walking robot prototype

Figure 2.6: Shanjing Power Supply, input:100-240 Volts and 1.2 A 50/60Hz, output:5V 7A

2.3 Servo-motor Drive Board (Servo Shield)

Due to high initial current of the motors which leads to drop of voltage in circuits and also safety (the motor drive shields may be burnt because of high current and heat), the motors should be fed separately from the other circuits. A servo-motor shield (board) is designed to connect all the servo-motors on a board and connect the board to a proper power source. The designed board selects the signal wire of servos and connects them to yellow wires to be connected to a main board to get controlling signals of motors.

2.4 Power Supply

An AC adapter is used to supply sufficient current to 18 servo-motors. Early experiments have shown that maximum initial current of servo-motors are around 500 mA at 6 Volts. Therefore, the **Shanjing Power Supply** 5V 7A adapter shown in figure 2.6 is chosen considering the safety margins and nominal working points. It works with 100-240 Volts and 1.2 A 50/60Hz input.

2.5 Low Level control of the Robot

The aim is to establish a real-time connection between the robot and MATLAB to implement the results for verification of formulation on an experimental model. MATLAB works fine with serial connection, but there is a challenging part. Each serial connection (USB or RS232) can send and receive one signal over **TX** and **RX** of serial connection protocol. Therefore, to control SiWaReL prototype, 18 serial connection would be required. However, this is not possible with regular Personal Computers or Laptops. There are some boards which are compatible with MATLAB for controlling real-time like National Instruments products, but they are expensive to be bought by personal budget.

One of the reasonable answers for connection challenge is to use a low level control architecture for tasks such as managing transmitting and receiving signals, sending the proper Pulse With Modulation (PWM) to all servomotors simultaneously. Then MATLAB can be used for a higher level control.

A board, which is an AVR microcontroller ATmega2560 base board, is used for low level control. It can control servo motors directly and also connect to a PC. The micro controller on the board should be programmed in a way to continuously read the serial port, and ,based on the received data from MATLAB, send the specified

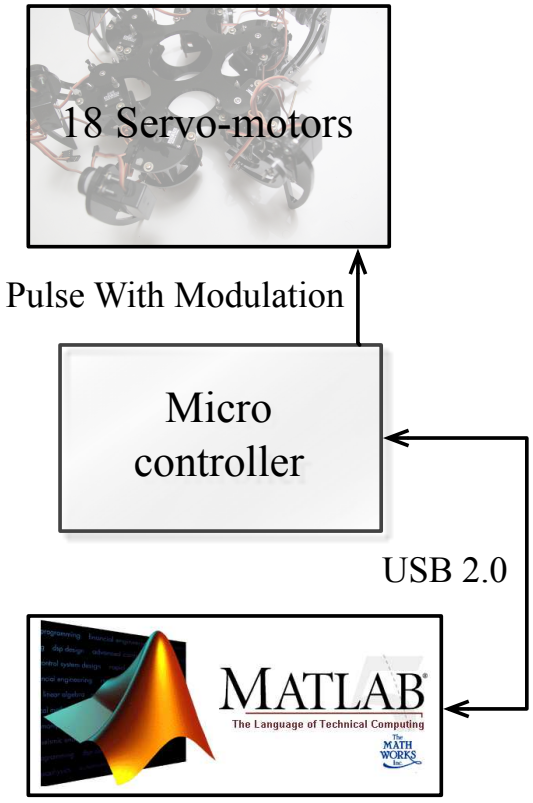


Figure 2.7: The diagram shows how to implement the real-time connection between the robot and MATLAB.

PWMs to servo-motors.

The designed servo shield connects the control wires of servo-motors to the main control board and also connects the servo-motors to AC adapter power supply. Main board micro controller is programmed to read serial port and send the proper signal to servo-motors. The main board itself is supplied with the voltage source of the USB as it is connected to the computer.

2.6 Summary

The prototype of SiWaReL robot is built and its procedures are defined in this chapter. The experimental challenges are defined and discussed. The real-time connection

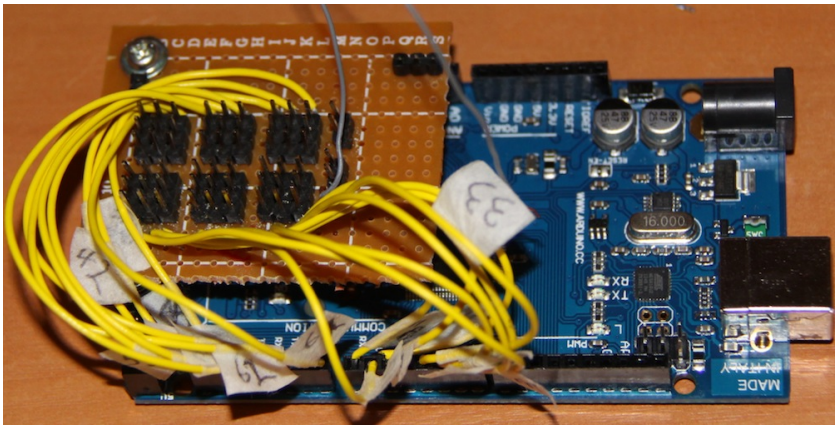


Figure 2.8: Serial Control board and servo shield

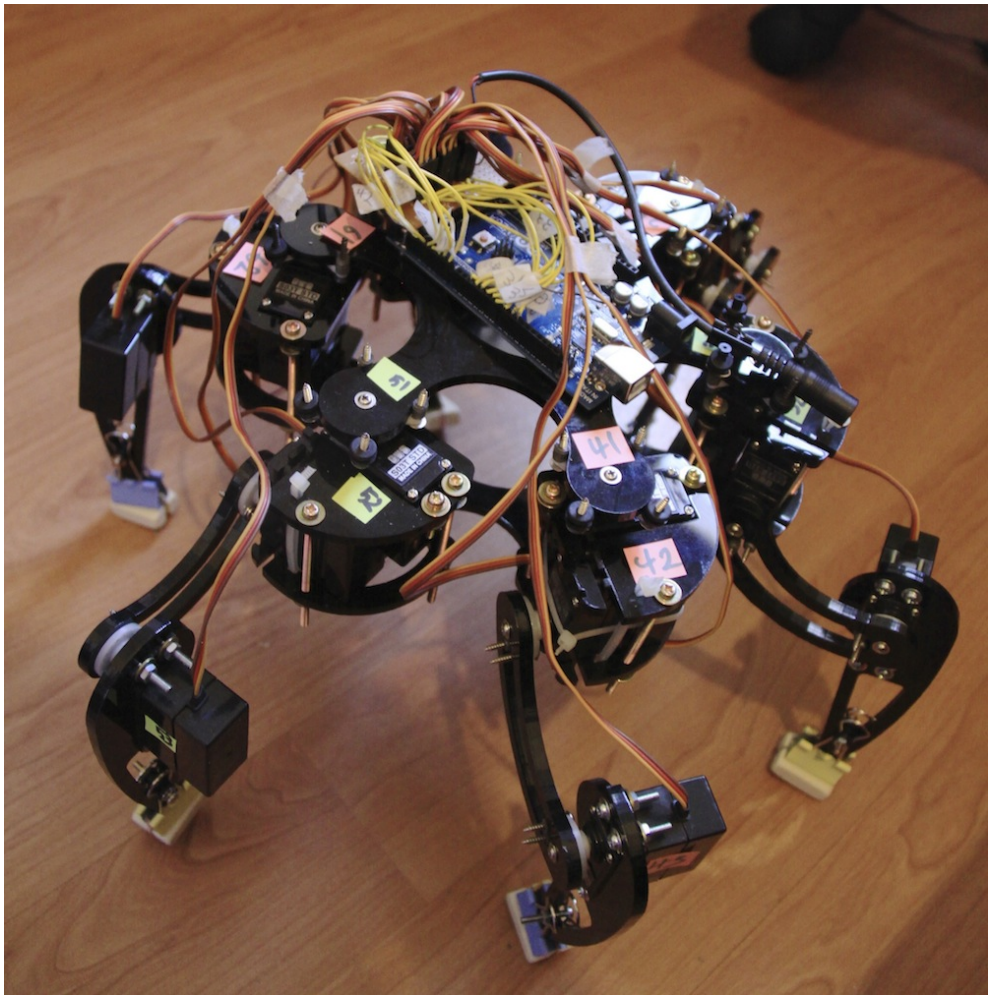


Figure 2.9: SiWaReL prototype with real-time connection to MATLAB.

between the prototype and MATLAB on a computer is established and the next step is to send proper joint angles arrays with respect to time, and to servo-motors through the designed and implemented interface boards.

Chapter 3

Modular View Kinematic Analysis of Hexapod

3.1 Introduction

Multi-Legged robot locomotion has been such a keen interest over the years to the researchers because of the advantages of the superior mobility in irregular terrain and the less hazardous influences on environment comparing with the wheeled robots. A multi-legged robot possesses a tremendous potential for maneuverability over rough terrain, particularly in comparison to conventional wheeled or tracked mobile robot. It introduces more flexibility and terrain adaptability at the cost of low speed and increased control complexity [22]. The kinematic properties of a six-legged robot can significantly influence locomotion procedure. A hexapod motion analysis is a complex combination of kinematic chains. Open chains when legs are in swing phase and closed chains when in stance phase with the trunk body. Lilly and Orin [23] treats a walking robot as a multiple manipulators (i.e. legs) contacting an object, which is the trunk body. Wang and Din [24] analyzed a radial symmetric hexapod kinematic and gait

analysis through a manipulation view by finding closed loops assuming the trunk is parallel to the ground and they did not consider the tilt of the trunk. Shah, Saha and Dutt [25] modeled legged robots as combination of floating-base three-type systems as kinematic modules where each is a set of serially connected links only. They used this idea for kinematic analysis of a biped and quadruped robots. In this chapter, we are using this idea for solving inverse kinematic problem of a radial symmetric six-legged robot. In this kind of hexapod robot, each leg has a different coordinate frame orientation compared to the other legs unlike rectangular hexapods which two sets of legs are oriented as two parallel sets in sides of the rectangular trunk. So their gait analysis and legs behavior are, kind of, different from each other in formulation. Here, we are going to propose a mobile view to solve the inverse kinematic problem of a six-legged robot assuming that the trunk has its 6 degrees of freedom. After solving the inverse kinematic problem, trajectory of each leg for gait analysis is the main problem as how to perform a swing step. For smooth walking, a typical swing cosine function [26, 27] is used and analyzed for tripod and wave gait.

3.2 Kinematic Analysis

The Hexapod prototype, which we are working on, has 3 Degrees of freedom for each leg, totally 18 degrees of freedom. Its a Google SKPR bot design, a biologically inspired design based on spiders anatomy. Each leg has three servomotors, which are modeled as 3 revolute joints as shown in figure 3.2.

3.2.1 Inverse Kinematic of Hexagonal Hexapod Robot

In locomotion analysis the problem is to find out how to assign the joint variables to move in the desired way, i.e. to find joint variables in terms of trunk configuration.

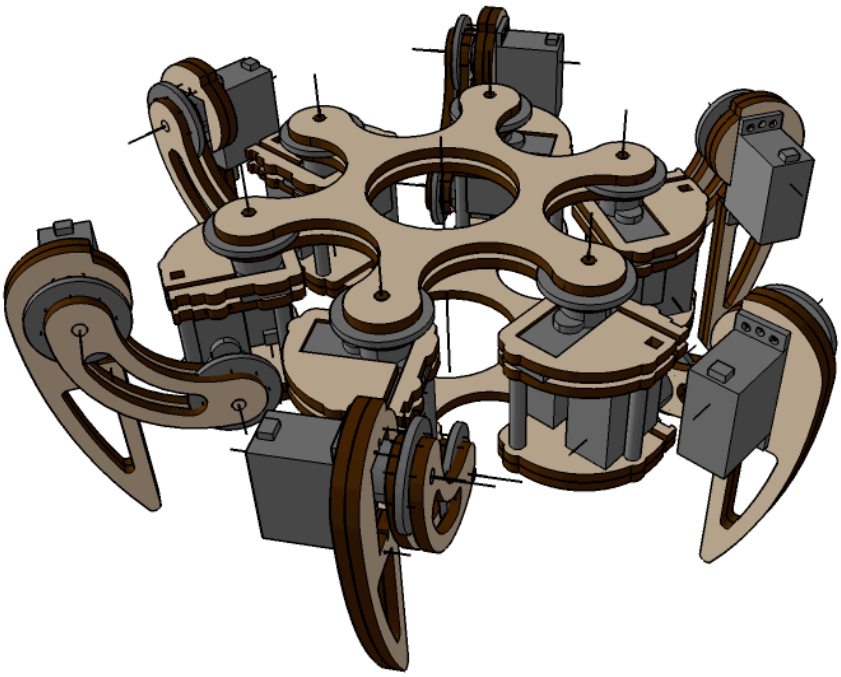


Figure 3.1: 18 DoF SKPRbot six-legged robot design

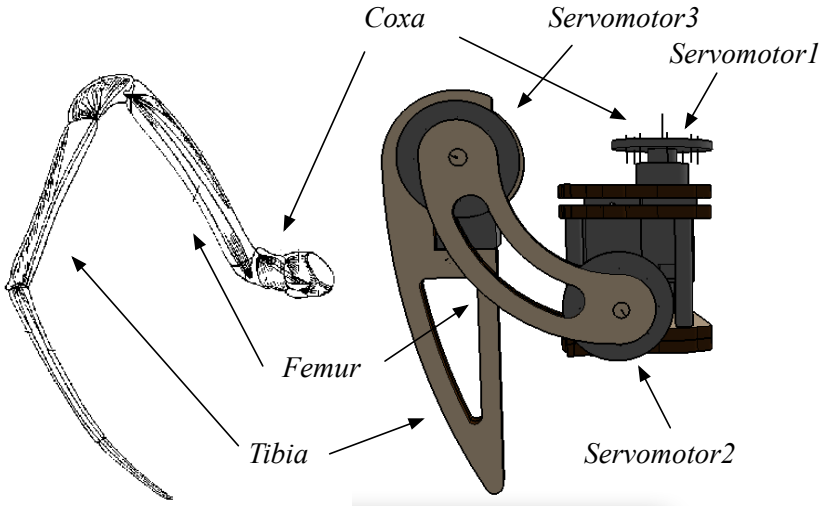


Figure 3.2: The hexapod leg design inspired by a spider leg

First of all, we simplify the architecture of the robot in 7 modules, a hexagon trunk and 6 limbs between the trunk and ground as shown in figure 3.3. Two coordinate frames are assigned. The First is $\{O\}$ on the ground, and the second one is the trunk, $\{O'\}$. 6 degrees of freedom (DoF) is considered for the trunk as shown in figure 3.3, 3 DoFs are axial movement in X', Y' and Z' direction and other 3 DoFs are rotational movement around X', Y' and Z' which are Roll, Pitch and Yaw respectively. Forward motion is considered as in the direction of X'.

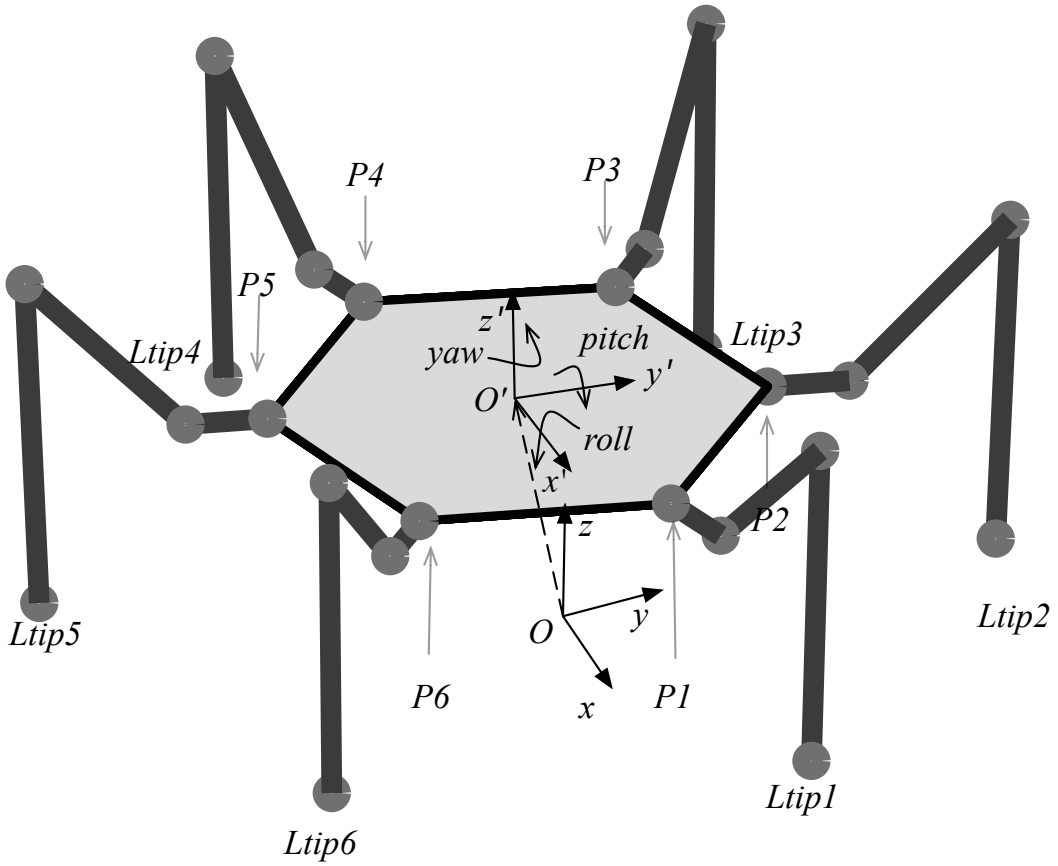


Figure 3.3: Hexagonal hexapod coordinate frame assignment, ground frame O and trunk frame O'

The main idea of solving inverse kinematic of this robot came from a modular view of floating trunk and serial kinematic chains [25]. To obtain the kinematic chain and find out the joint variables, we define a homogenous transformation matrix to

transform the Legs' tips from ground coordinate frame to trunk coordinate frame. This transformation matrix can be written as below:

$$ {}^{O'}_{O}T = \begin{bmatrix} R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) & -OO' \\ 0 & 1 \end{bmatrix} \tag{3.1} $$

R_z, R_y and R_x are rotational transformation matrices around Z, Y and X respectively and OO' is the distance from O' to O. In coordinate frame O', $P_1..P_i..P_6$ are the corners of the trunks hexagon and always have constant value in $\{O'\}$ and L_{tip_1}.. $L_{tip_i}..L_{tip_6}$ are legs tips position in $\{O\}$. Here, the Legs tip positions are taken from $\{O\}$ to $\{O'\}$ using defined transformation matrix.

$$ L'_{tip_i} = {}^{O'}_{O}T L_{tip_i} \tag{3.2} $$

$$ L_{tip_i} = \begin{bmatrix} x_{tip_i} \\ y_{tip_i} \\ z_{tip_i} \end{bmatrix} \tag{3.3} $$

And now in $\{O'\}$ the inverse kinematic can be solved from (3) for each leg.

$$ \begin{bmatrix} x_{lt_i} \\ y_{lt_i} \\ z_{lt_i} \end{bmatrix} = L'_{tip_i} - P_i \tag{3.4} $$

$$ P_i = \begin{bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \end{bmatrix} \tag{3.5} $$

$$
\begin{aligned}
x_{lt_i} = \quad & x_{tip_i} cos\theta_y cos\theta_z \\
& + y_{tip_i}(cos\theta_x sin\theta_z + cos\theta_z sin\theta_y sin\theta_x) \\
& + z_{tip_i}(sin\theta_x sin\theta_z - cos\theta_x cos\theta_z sin\theta_y) \\
& - OO'_x + P_{ix}
\end{aligned}
\tag{3.6}
$$

$$
\begin{aligned}
y_{lt_i} = \quad & - x_{tip_i} cos\theta_y sin\theta_z \\
& + y_{tip_i}(cos\theta_x cos\theta_z - sin\theta_y sin\theta_x sin\theta_z) \\
& + z_{tip_i}(cos\theta_z sin\theta_x + cos\theta_x sin\theta_y sin\theta_z) \\
& - OO'_y - P_{iy}
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
z_{lt_i} = \quad & x_{tip_i} sin\theta_y - y_{tip_i} cos\theta_y sin\theta_x \\
& + z_{tip_i} cos\theta_y cos\theta_x - OO'_z - P_{iz}
\end{aligned}
\tag{3.8}
$$

3.2.2 Forward and Inverse Kinematic Analysis of one Leg

Each leg can be seen as a serial manipulator where its base is fixed on the trunk and its end point is on the ground or on its swing path. For forward kinematic analysis, using Denavit-Hartenberg parameters of one leg [26], we can write transformation matrix of each joint, based on frames shown in figure 3.4 and table one. Three joints and five frames are defined from the initialized frame to the end point of the leg.

The homogeneous transformation matrices of legs links based on DH parameters in table 1 are presented as below:

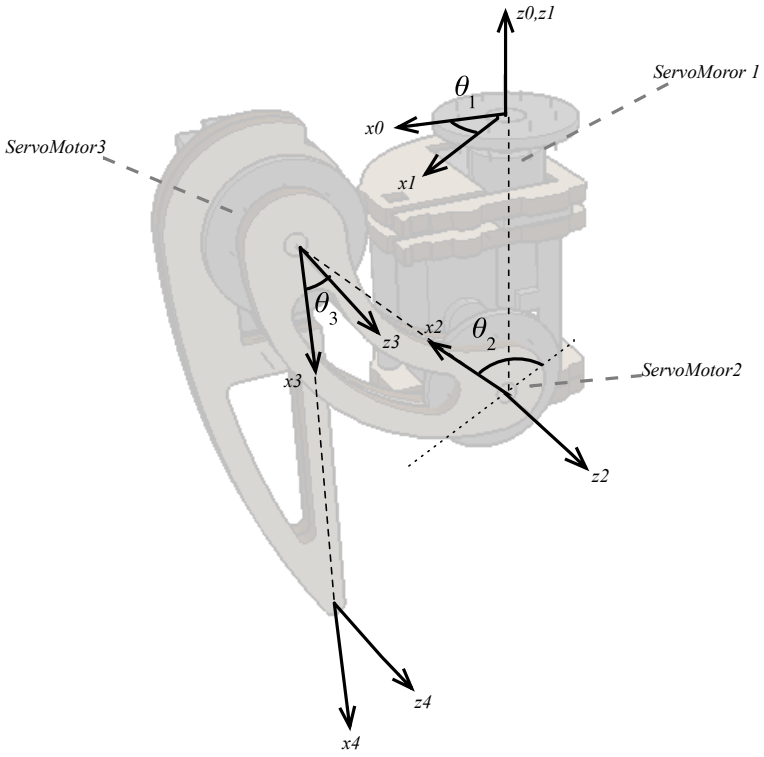


Figure 3.4: Link-frame assignments for finding Denavit Hartenberg parameters

$$
{}^{0}_{4}T_i = \begin{bmatrix} c_{12_i}c_{1_i} & -s_{12_i}c_{1_i} & -s_{1_i} & c_{1_i}(l_0 + l_1c_{2_i} + l_{2_i}c_{23_i}) \\ c_{12_i}s_{1_i} & -s_{12_i}s_{1_i} & c_{1_i} & s_{1_i}(l_0 + l_1c_{2_i} + l_2c_{23_i}) \\ -s_{23_i} & -c_{23_i} & 0 & -l_1s_{2_i} + l_2s_{23_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}
$$

where c_1 , c_{12} , c_{23}, s_1 , s_{12} and s_{23} stand for $cos\theta_1$, $cos(\theta_1 + \theta_2)$, $cos(\theta_2 + \theta_3)$, $sin\theta_1$, $sin(\theta_1 + \theta_2)$ and $sin(\theta_2 + \theta_3)$ respectively. ${}^{0}_{4}T_i$ transforms points from end point which is the leg's tip to the base coordinate frame. The position of the leg tip in $X'Y'Z'$ coordinate frame can be found using homogeneous transformation matrix from base coordinate frame to endpoint coordinate frame. The reason that O' is defined adjacent

to O is the vertical distance between the shaft of servomotors number 1 and 2.

$$\begin{bmatrix} x_{lt_i} \\ y_{lt_i} \\ z_{lt_i} \\ 1 \end{bmatrix} = {}^{0}_{4}T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.10}$$

$$x_{lt_i} = c_{1_i}(l_0 + l_1 c_{2_i} + l_2 c_{23_i}) \tag{3.11}$$

$$y_{lt_i} = s_{1_i}(l_0 + l_1 c_{2_i} + l_2 c_{23_i}) \tag{3.12}$$

$$z_{lt_i} = -l_1 s_{2_i} + l_2 s_{23_i} \tag{3.13}$$

where x_{lt} , y_{lt} and z_{lt} are leg tip position in $\{O'\}$ and θ_1, θ_2 and θ_3 are joint angels. Based on 3.3 Inverse kinematic equations for one leg can be written as:

$$\theta_{1i} = arctan2(y_{lt_i}, x_{lt_i}) \tag{3.14}$$

$$d_i = \sqrt{(x_{lt_i} - l_0 c_1)^2 + (y_{lt_i} - l_0 s_1)^2 + z_{lt_i}^2} \tag{3.15}$$

$$B_i = acos(\frac{d_i^2 + l_1^2 - l_2^2}{2 l_1 d_i}) \tag{3.16}$$

$$\theta_{2i} = asin(\frac{-z_{lt_i}}{d_i}) - B_i \tag{3.17}$$

$$C1_i = acos(\frac{l_1 sin B_i}{l_2}) \tag{3.18}$$

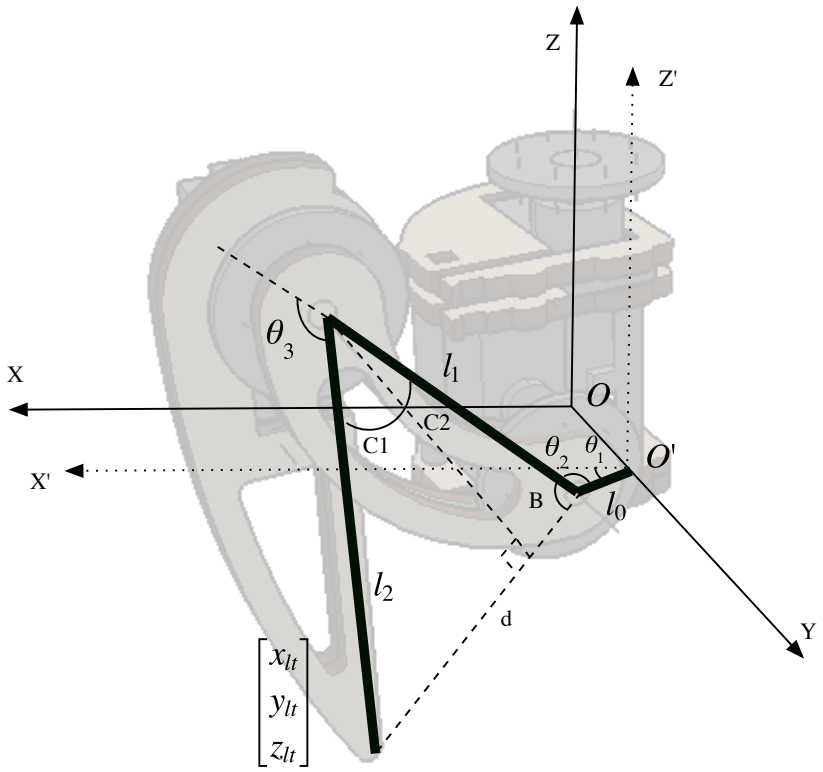


Figure 3.5: A 3 DoF hexapod leg design link assignment and parameters for inverse kinematic analysis of one leg

Table 3.1: Denavit-Hartenberg parameters of 3 DoF hexapod's leg

Link	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-\pi/2$	l_0	0	θ_2
3	0	l_1	0	θ_3
4	0	l_2	0	0

$$C2_i = \frac{\pi}{2} - B_i \tag{3.19}$$

$$\theta_{3i} = \pi - C1_i - C2_i \tag{3.20}$$

3.3 Gait Analysis of Hexapod Robot

Six legs are moving together to form a walking gait. Every walking gait can be simplified to some similar rules for every leg namely step. By applying these algorithms to each leg, walking can be achieved. In gait analysis, the leg tasks can be classified as two phases, stance and swing phases. When the robot is moving on desired trajectory i.e, $P_x(t)$, $P_y(t)$, $P_z(t)$, $\theta_x(t)$, $\theta_y(t)$ and $\theta_z(t)$, some legs on the ground are pushing to move the trunk in forward direction while the other legs are getting into new position. These two leg phases are augmented to L_{tip_i}. Inverse kinematic in each step can be found by solving trajectory using Leg tips on the ground through (3.6) to (3.8) and (3.14) to (3.20). While the robot is moving, some legs swing forward to get into the new position for the next step. It is important for the leg not to impact

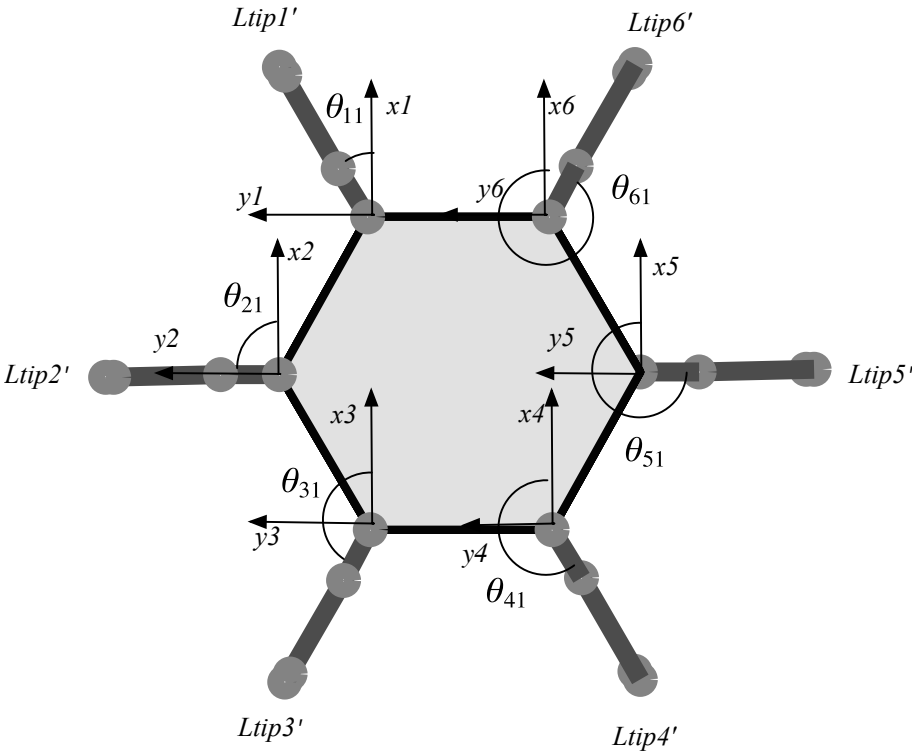


Figure 3.6: 18 DOF SKPRbot six-legged robot design

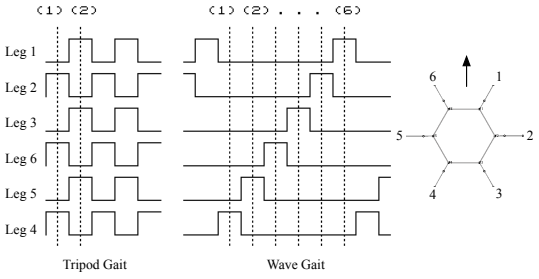


Figure 3.7: Tripod Gait and Wave Gait signals sequences

the ground. The velocity at the start and end of swing phase should be zero, so to get also a smooth actuation signal a path is defined for $i \in (swing\ legs)$ to go into a new position.

$$x_{tip_i} = 2\dot{P}_x \delta t_{step}(1 - cos(\frac{\pi t}{\delta t_{step}})) \tag{3.21}$$

$$y_{tip_i} = 2\dot{P}_y \delta t_{step}(1 - cos(\frac{\pi t}{\delta t_{step}})) \tag{3.22}$$

$$z_{tip_i} = h(1 - cos(\frac{\pi t}{\delta t_{step}})) \tag{3.23}$$

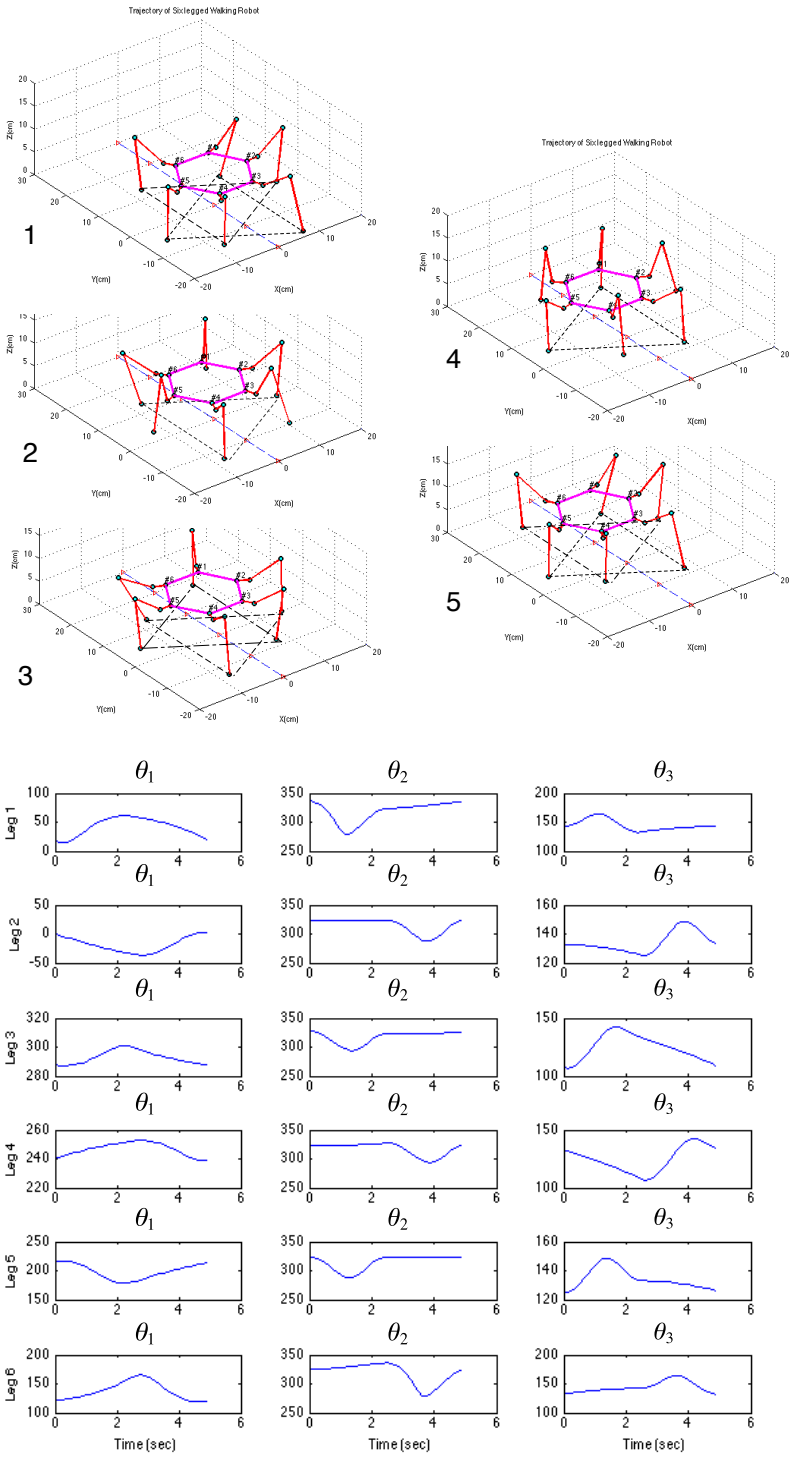
$\dot{P}_x$ is vertical speed of trunk in x direction, $\dot{P}_y$ is the speed of robot's trunk in y direction, h is the height of each step in swing phase and δt_{step} is the time duration of each step.

3.3.1 Simulations of Gait Analysis

Two walking gaits, wave and tripod gait, have been studied and simulated using presented formulation. The joint values with respect to time are found using leg tips and body position with inverse kinematic formulations.

In tripod gait, for example, two equilateral triangles are defined, one for standing legs and one for another swinging legs. The standing legs are on the ground and form a triangle. When the robot is going forward on standing legs the other triangle (the other three legs' tip forms) is moving forward above the ground to get into new position, i.e. swing phase as shown in figure 3.7. The results of inverse kinematic for tripod gait are shown in figure 3.8.

In wave gait, robot moves its legs one by one to get the highest stability margin but so slower, as shown in figure 8. Results and the simulation are in figure 3.9.



.

Figure 3.8: Joint space variables of the hexapod robot in tripod gait through one complete step (left), Tripod walking gait simulated leg sequences in one complete step. Triangles are grounded leg tips (right).

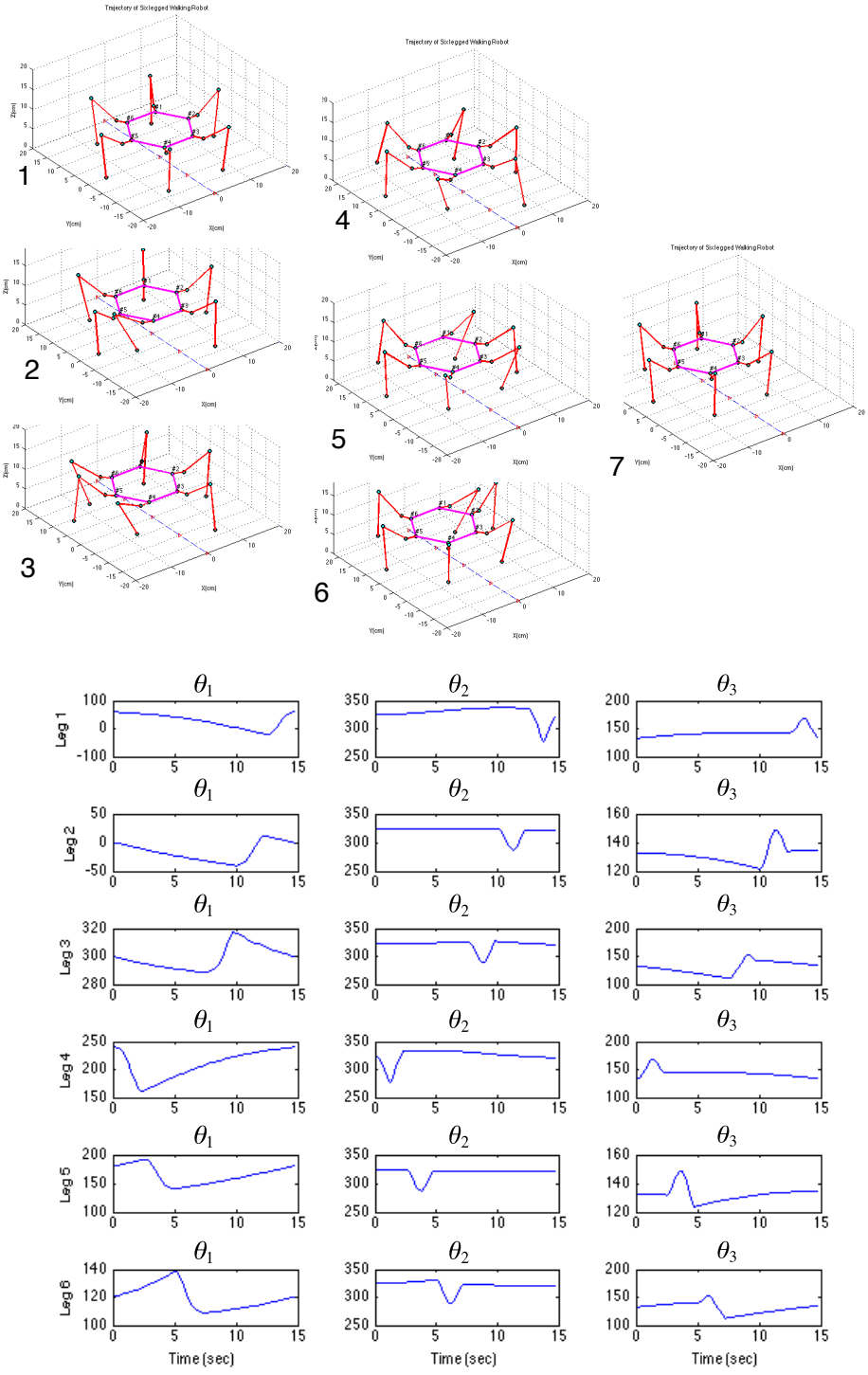


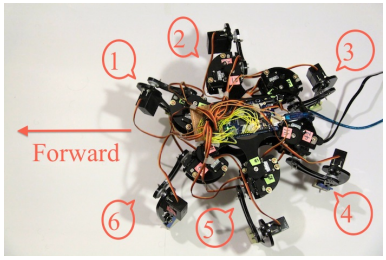
Figure 3.9: Joint space variables of the hexapod robot in wave gait through 1 step (Left), Wave walking gait simulated leg sequences in one complete step. (Right).

3.3.2 Implementation Gait Analysis and Inverse Kinematic Formulations on SiWaReL Prototype

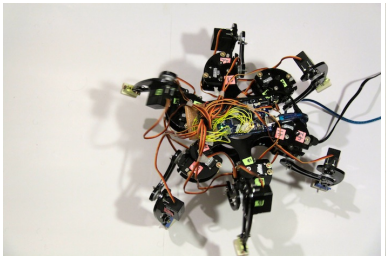
SiWaReL prototype is used to verify the formulation. In previous section, the inverse kinematic formulation is tested with two typical gaits, tripod and wave gait. In both gaits, the related joint-time arrays are generated. Using these specific values and sending them to SiWaReL hexapod prototype, the results of walking can be seen with feed forward control.

Therefore, by sending the gait analysis results, i.e., joint value to the prototype, it can be seen how it walks. The connection between the robot and MATLAB is established and the sampling of joint values is done every 10 miliseconds which results in smooth walking of the robot.

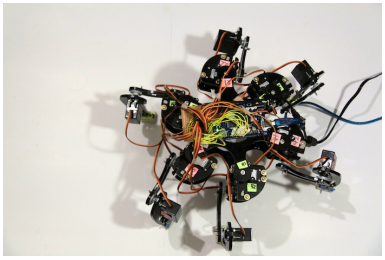
As it is shown in figure 3.10 and 3.11, the robot walks without any problem as it was predicted in the simulations. The robot walks simultaneously as MATLAB sends joint values. The micontroller which manages the connection between the MATLAB and the robot is programmed considering the stability in cases that MATLAB is busy or sends signal by delay. This feature provides robust connection for the real time control.



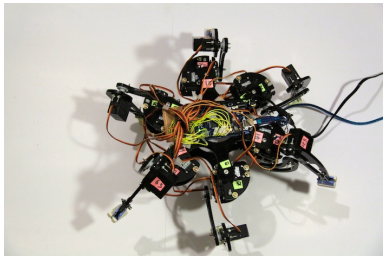
(a) Robot at rest



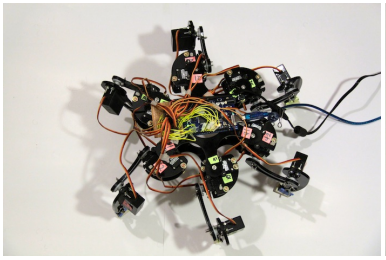
(b) Legs 1,3 and 5 are moving into new position.



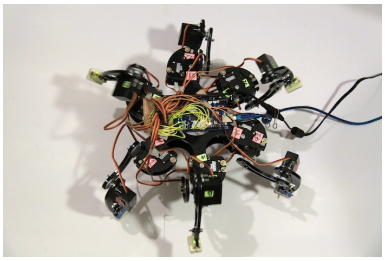
(c) Legs 1,3 and 5 are in new position.



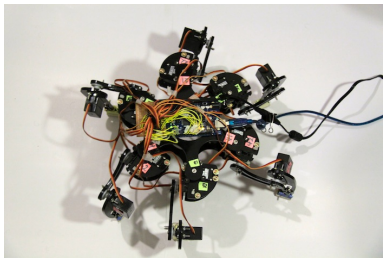
(d) Legs 2,4 and 6 are moving into new position.



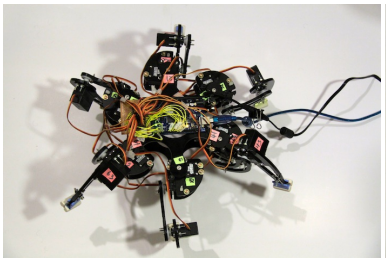
(e) Legs 2,4 and 6 are in new position.



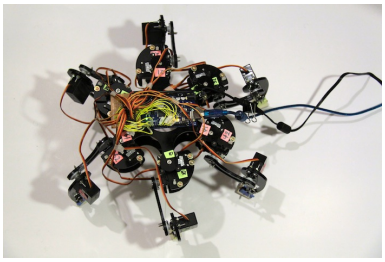
(f) Legs 1,3 and 5 are moving into new position.



(g) Legs 1,3 and 5 are in new position.

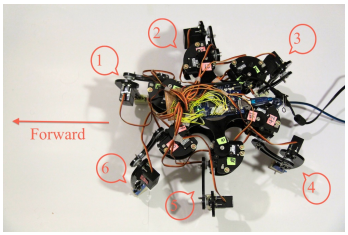


(h) Legs 2,4 and 6 are moving into new position.

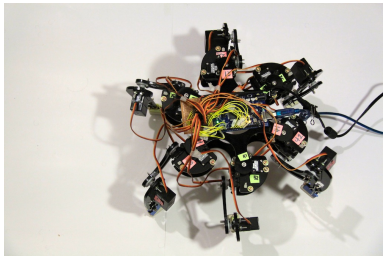


(i) Robot is standing.

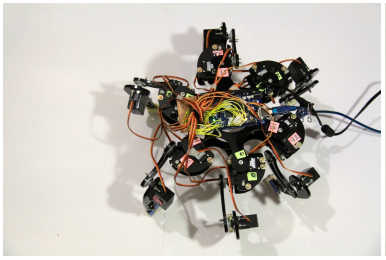
Figure 3.10: Tripod gait implementation on SiWaReL prototype in 2 steps



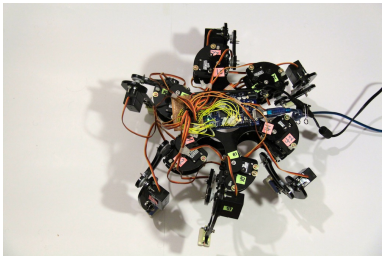
(a) Robot in Rest



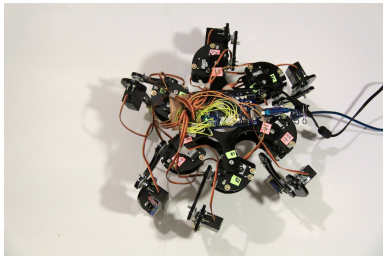
(b) Leg 4 is moving.



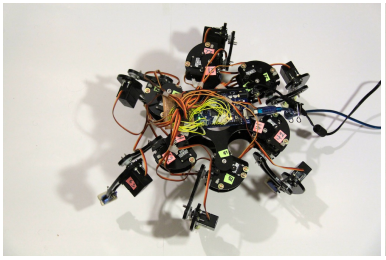
(c) Leg 4 is in new position .



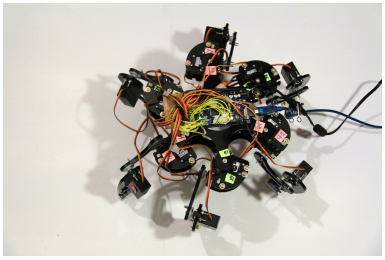
(d) Leg 5 is moving.



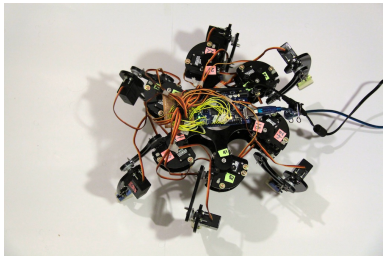
(e) Leg 5 is in new position.



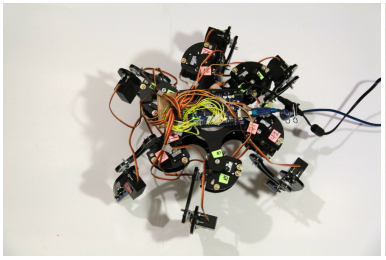
(f) Leg 6 is moving.



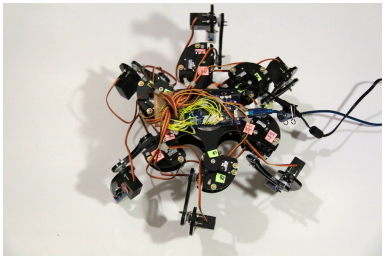
(g) Leg 6 is in new position.



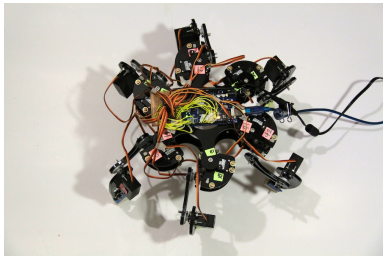
(h) Leg 3 is moving.



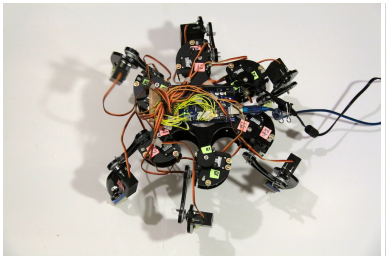
(i) Leg 3 is in new position.



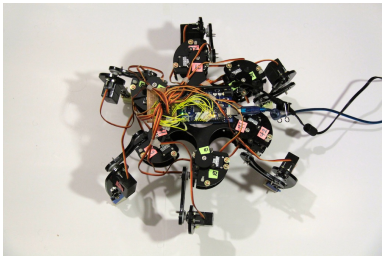
(j) Leg 2 is moving.



(k) Leg 2 is in new position.



(l) Leg 1 is moving.



(m) Leg 1 is in new position.

Figure 3.11: Wave gait implementation on SiWaReL prototype in one step

3.4 Summary

In this chapter, inverse kinematic formulation of a radial symmetric (hexagonal) hexapod has been presented considering 6 degrees of freedom for the trunk. Trajectories of two gaits are solved through the presented formulation and simulated with smooth actuation signals. It is shown that a modular view for solving inverse kinematic problem for this kind of robot simplifies the complexity of different orientation of legs and gait analysis can be implemented as other robots with a general algorithm to each leg despite the orientation of its legs coordinate frames. The results of gait analysis and kinematic formulation have been implemented on SiWaReL hexapod prototype and it is shown that the formulation works fine with an experimental model.

Chapter 4

Dynamic Modeling of SiWaReL Hexapod Robot

In order to control the robot, a model of the system is required. At first, it was planned to do an online training using Reinforcement Learning techniques for the robot to learn how to walk. A prototype has been built for this purpose as the studies and tests have been done. It is shown that this has some difficulties which could not be neglected in this project. As it has been seen in the literature, some projects using high quality servo motors and interfaces have faced burning circuits and motors before the learning iterations were done completely [28]. Hence, it is not affordable to do online training with the prototype. The new approach was changed to do the training offline with an accurate model, and then to implement the results on the prototype to validate the learning results.

Dynamic modeling of robotic systems is a wide field of research and lots of approaches and engines are developed to prepare an environment to model the dynamic behaviors of robots like forward and inverse dynamic analysis of robots and interactions with the ground, objects and other robots. There are some open sources

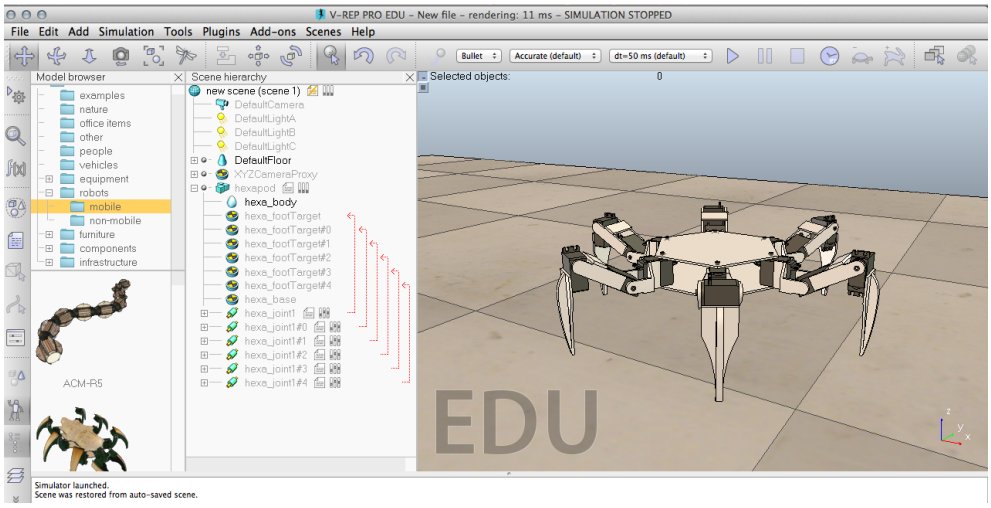


Figure 4.1: V-rep robot simulation environment. A hexapod walking robot is simulated here from its own library and walking algorithms.

and commercially available programs that are developed for dynamic modeling and simulation of different kind of robots with different goals. In the next section some of these programs which have the capability of a hexapod model and implementing different control algorithms are discussed.

4.1 Hexapod Robot Model in Dynamic Simulation Environments

4.1.1 V-rep, Coppelia Robotics

V-rep is an open source free program which has a great library of different kinds of robot, from snake robots, humanoid robots to hexapods and wheeled robots. The main advantage of this environment is its user friendly environment and simplicity of the procedure of simulating and designing; it can also be installed on any operating system with no cost.

Figure 4.1 shows the environment of V-rep software and a hexagonal hexapod

available from its library. The algorithms are written in v-rep's own programming language which is a C based language but not exactly like C. There is an online catalog for its codes. This environment is good for this project, but it requires learning a new programming language to formulate the control algorithm with. However, V-rep has the ability to connect with programs like MATLAB over remote API connection. Although remote API connection is feasible, as it was tried, it has been seen that it does not work with an acceptable speed which is necessary for real time online training. So V-rep was not the proper simulation environment for this online training of walking problem.

4.1.2 Webots, Cyberbotics

Webots is an efficient robot simulator with useful features like having a huge library of different sensors, actuators and also robots. It provides the capability of control robots using MATLAB *.m* files directly. This simulator has the model of existing available prototype robots including epuck, Nao, DARwin-OP and KUKA youBot. It is a commercially available program which also has a free 30 day trial. It can be installed on Windows, Mac OS and Linux operating systems. It is a proper software for those who want to implement something on these specific robots which are designed accurately and exist in Webots library. As it has been searched, hexagonal hexapods have not been modeled in Webots yet. Although a robot design user guid exists on Webots and we tried to design and build SiWaReL in Webots, the designing procedure in a new environment needs lots of time and the knowledge of that specific program. Therefore, despite the fact that Webots is a useful program for implementing control algorithms on a robot in dynamic simulation, it is not feasible to spend so much time and energy on designing a robot in Webots for this project.

There are lots of open source simulators that use engines like Open Dynamic

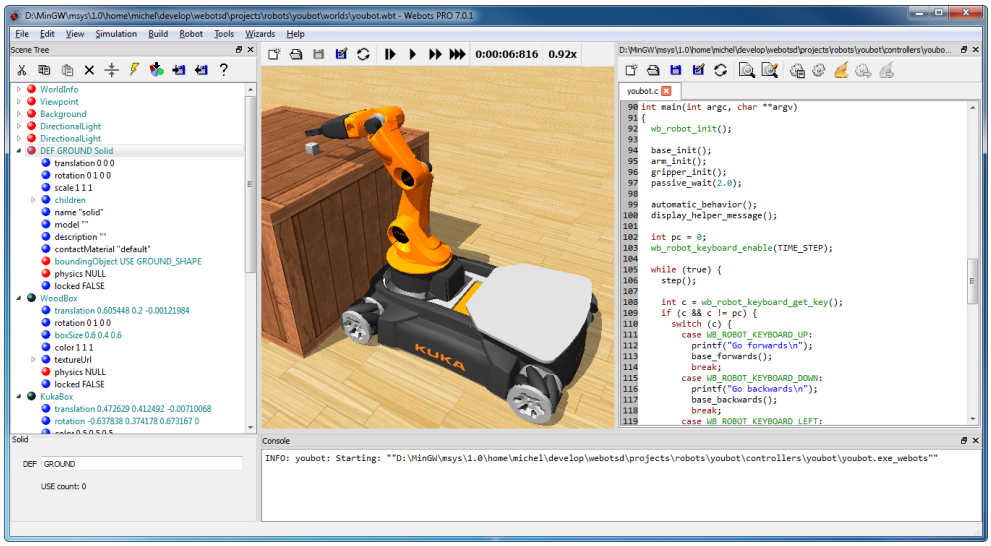


Figure 4.2: Webots robot simulation environment

Engine (ODE) which are designed for specific robots or with different kinds of robots. Gazebo robot simulator is said that is the most complete of its kind, but it does only work on Linux. There are other programs like ORCA simulator, Microsoft Robotic Developer Studio, RoboLogix, COSIMIR, Breve, LpzRobots, OpenHRP3, etc, that are useful for robotic applications and implementation in a simulation environment.

4.2 Dynamic Modeling of Hexapod Robot in MATLAB© SimMechanics

MATLAB SimMechanics is a simulink based simulation environment designed for modeling and simulation of dynamic systems using objective oriented methods. Systems can be modeled using predefined blocks such as bodies, joints, sensors, actuators, etc. The main advantage of this environment is that it builds a model which can be used in any part of MATLAB without any need of other specific program or remote connection.

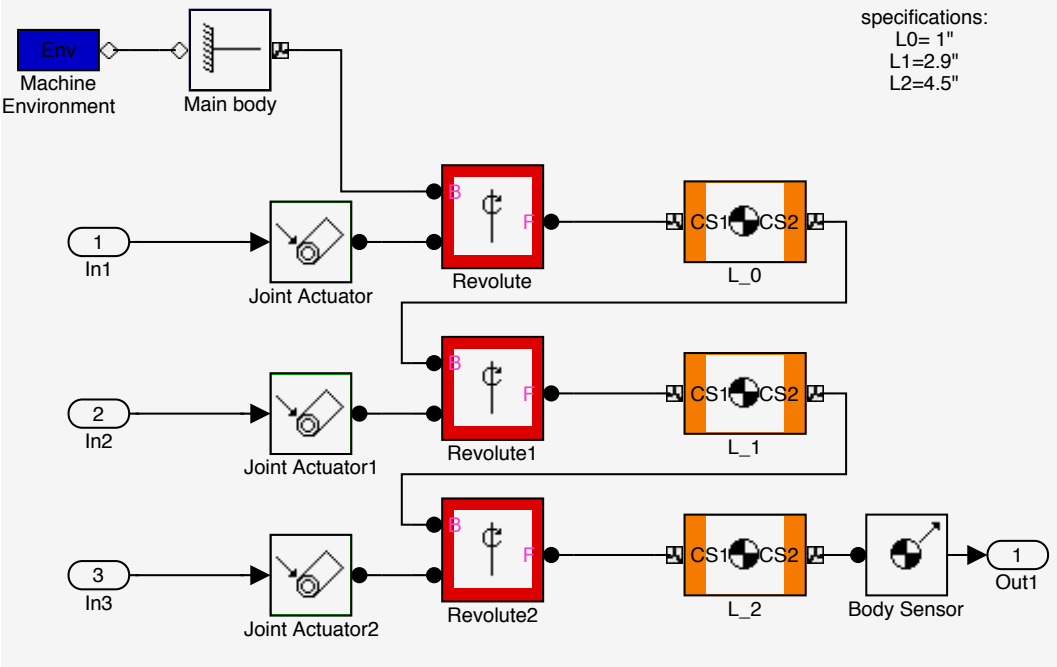


Figure 4.3: SimMechanics Model a biologically inspired 3 degrees of freedom Leg

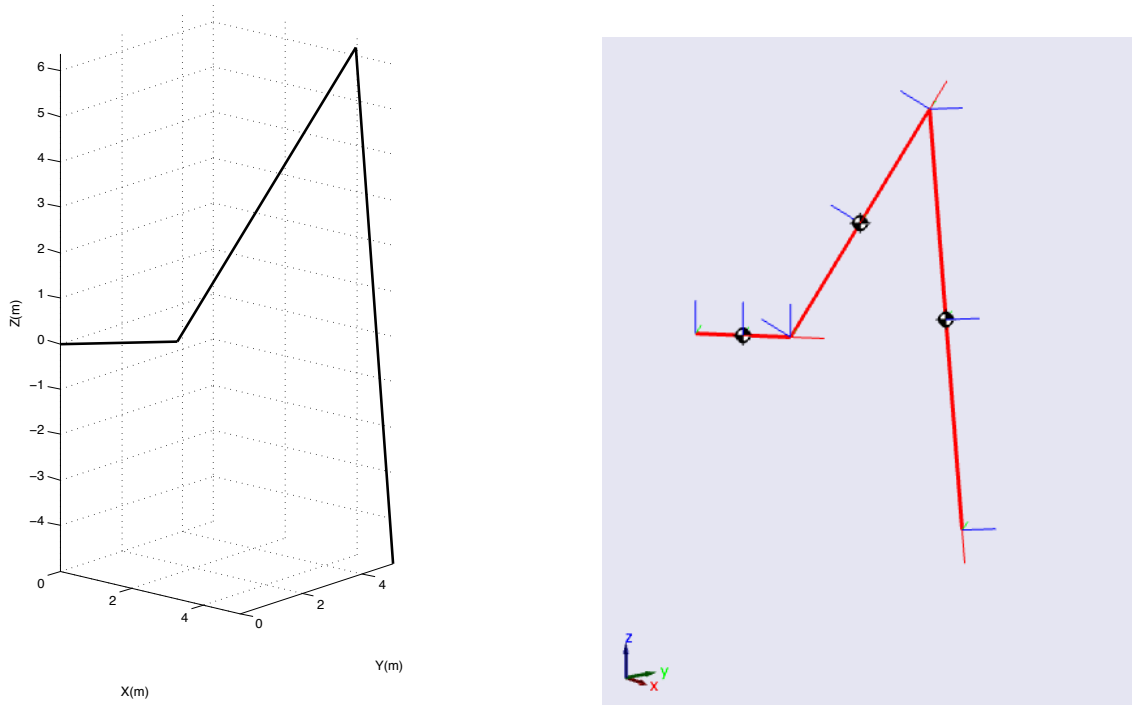
4.2.1 Leg Design

Based on design which is discussed in chapter 3, each leg has 3 degrees of freedom. Using 3 revolute joints and 3 links, a dynamic model of leg can be defined in SimMechanics. As the results show in figure4.4, this design matches the inverse kinematic analysis of the SiWaReL, so it is verified.

4.2.2 Hexapod Design

The whole robot designing is possible by connecting 6 legs to the main body. But as in every modeling environment, one of the challenges is to model the restrictions and limits. In this specific problem, the main challenging part is to model how the robot could interact with the ground and how this design would walk on the ground.

As study shows the ground contact can be modeled with a spring damper model. So in every leg tip, between the leg and the ground, there is a spring damper that acts



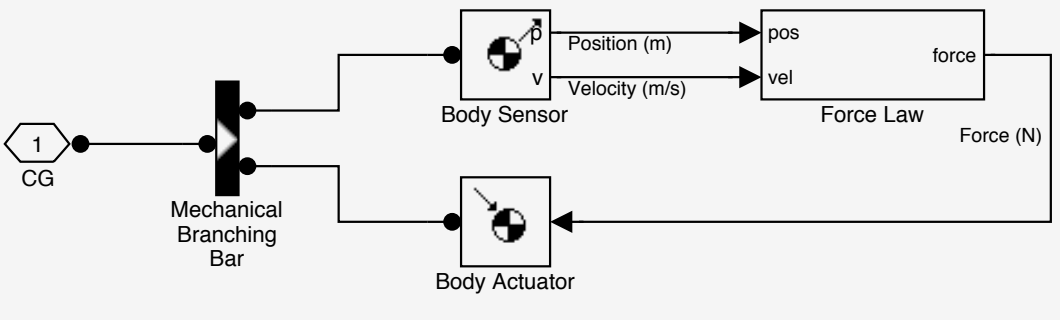
(a) Kinematic Model (b) Dynamic Model

Figure 4.4: Model of 3 DOF leg

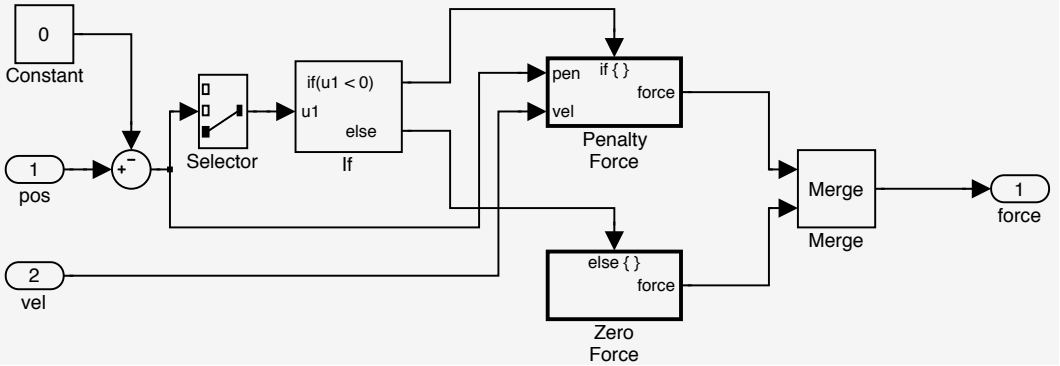
only if a leg touches the ground to model the resistance of the leg tip from inserting to ground. It should be noted that when the leg is on the transfer phase (i.e. swing phase), the spring damper should not have any effect on the leg. The ground contact force model is shown and described in figure 4.5. So based on ground contact, leg model will be developed to a new model that can interact with ground by contacting the tip of leg to *CG* port which is shown in figure 4.5. The complete leg design can be seen in figure 4.6.

So based on mentioned leg design, the whole robot model is built and the schematic is in figure 4.7. The main body (trunk) of the hexapod is connected to the ground frame with a custom 6 DoF joint which is not actuated. This joint is used to link the robot to the environment characteristics like gravity vector and the ground surface.

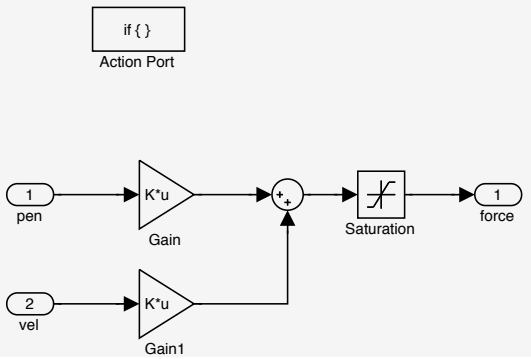
The simulation results show that the kinematic model and the dynamic model



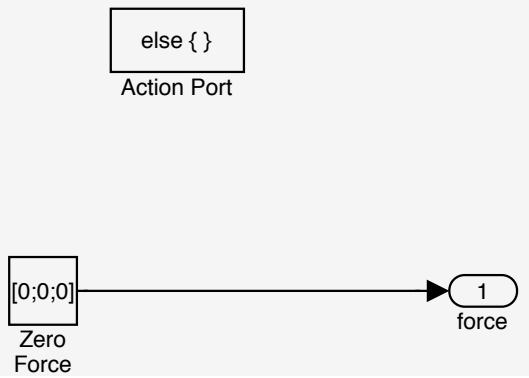
(a) Diagram of spring damper modeling for ground contact. CG links to tip of hexapod's leg.



(b) Force Law block which defines how ground applies force to the legs.



(c) Penalty force when leg is on the ground.



(d) Zero force when leg does not touch the ground.

Figure 4.5: Ground contact model in SimMechanics

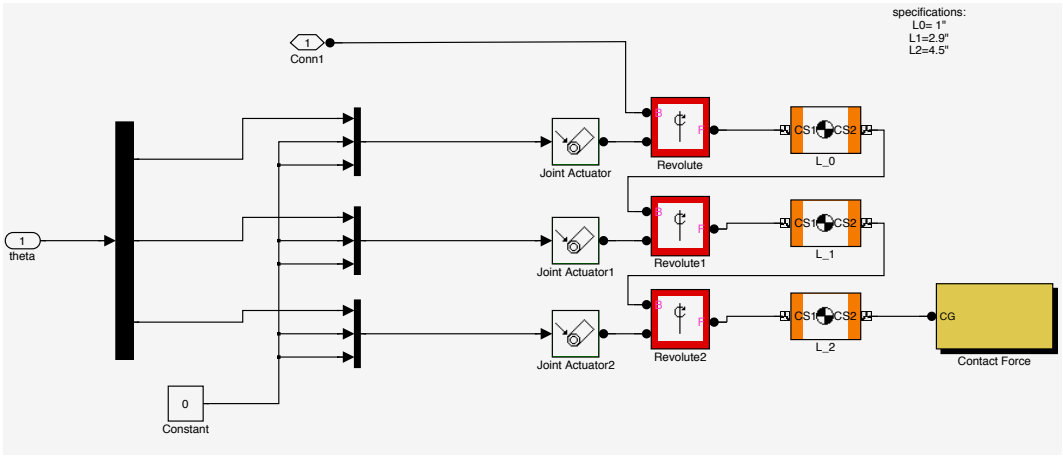


Figure 4.6: Complete model of 3 DOF Leg with ground contact model

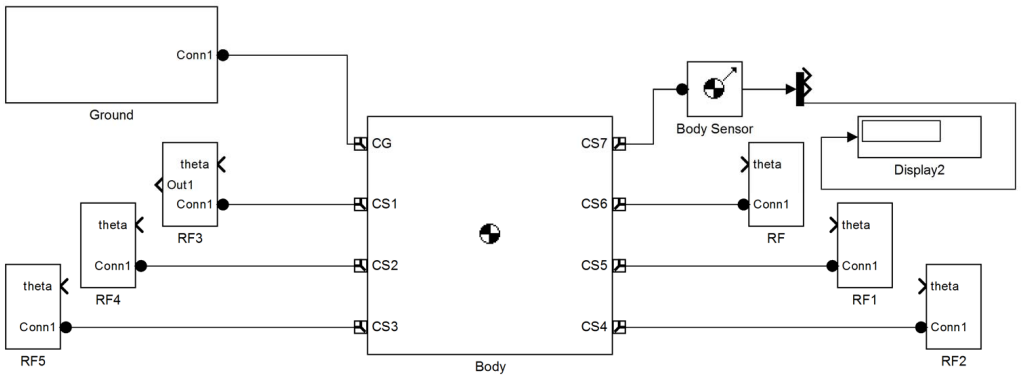
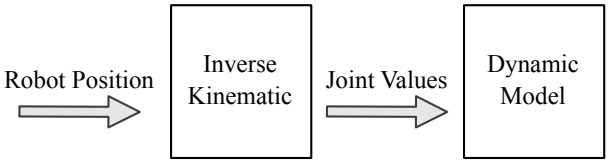
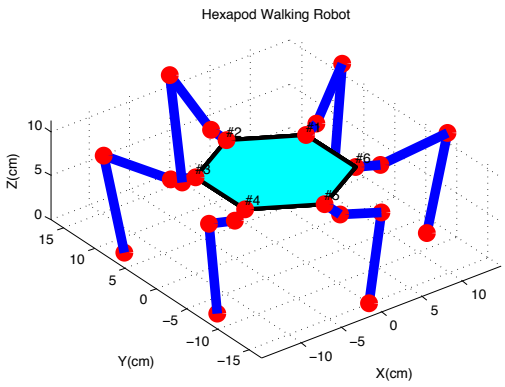


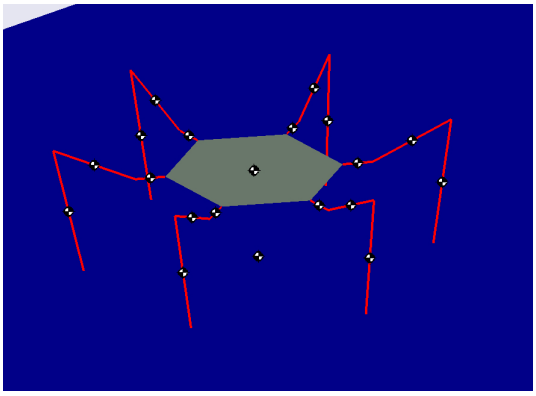
Figure 4.7: Hexapod robot design in SimMechanics



(a) The schematic of simulating dynamic model.



(b) Kinematic model of hexapod



(c) Dynamic model of hexapod

Figure 4.8: Dynamic modeling of six-legged walking robot in MATLAB Simulink, SimMechanics

match. An assumed robot position is solved through inverse kinematic formulation. Then the dynamic model is simulated with the results of inverse kinematic. Figure 4.8 shows the graphics of both kinematic and dynamic model.

4.3 Summary

In this section the dynamic model of a radially symmetric hexapod robot is developed in MATLAB SimMechanics which is a simulation environment based on MATLAB simulink simscape that uses objective oriented modeling to model dynamic systems in MATLAB. In Reinforcement Learning of walking state and actions of the robot are going to be simulated using the developed model. In the next chapter the walking learning of the robot will be discussed.

Chapter 5

Reinforcement Learning in Hexapod Walking

Reinforcement Learning (RL) techniques are interesting subjects in both control theory and cognitive sciences. In control theory, building a system that works completely perfect is quite difficult, and it is an exhaustive procedure when unexpected errors or disturbances affect the system. By building a system that learns how to accomplish a task on its own, there becomes no need to calculate and predict complex control algorithms. In cognitive sciences, the ability to learn is a core component of cognition. Reinforcement learning algorithm is one such simple learning algorithm. This section explores the ability of a robotic hexapod agent to learn how to walk, using only the ability to move its legs and tell whether the robot is moving forward. Therefore, the hexapod may be seen as an analog for a biological subject lacking all but the basic instincts observed in infants and having no external support or parental figure to learn from.

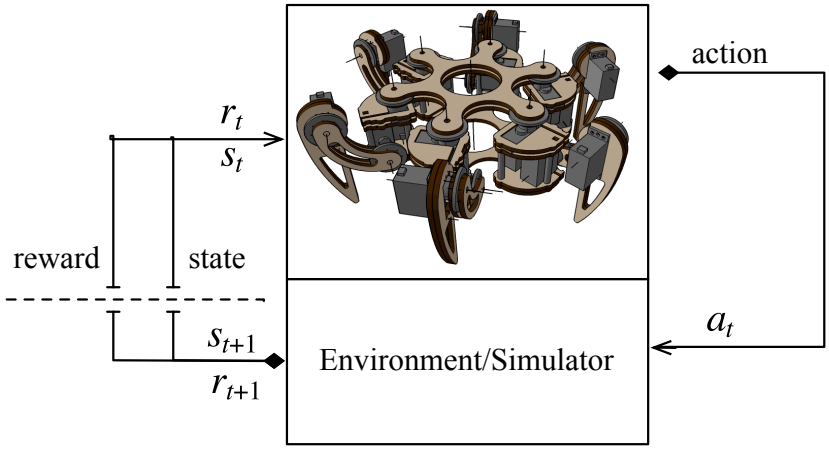


Figure 5.1: The diagram of reinforcement learning procedure. Actions taken by agent lead to certain states and reward. The goal is to find the best policy to take actions with the highest rewards.

5.1 Reinforcement Learning Problem Architecture

In supervised learning, the targets, i.e. right answers, are given to the agent as the training set. In some control problems, it is difficult or not feasible to define the exact correct supervision. For example, in hexapod walking, it is hard to define explicit supervision that the algorithm of learning is trying to mimic. In reinforcement learning instead of telling the exact supervision, only a reward function shows that the agent is doing well or not. So it will be learning algorithms job to find the best actions which lead to larger rewards. There will be no need to define input/output sets. This kind of learning focuses on online performance and the interaction between exploration and exploitation. The trade between these two has been one of the most challenges which have been studied in recent years [29].

It has been shown that reinforcement learning has different successful applications in autonomous systems and legged robot locomotion [30]. Q-learning is a simple approach of Reinforcement Learning which has been chosen for walking learning. To define the problem, here, firstly, actions and states of problem should be defined. A

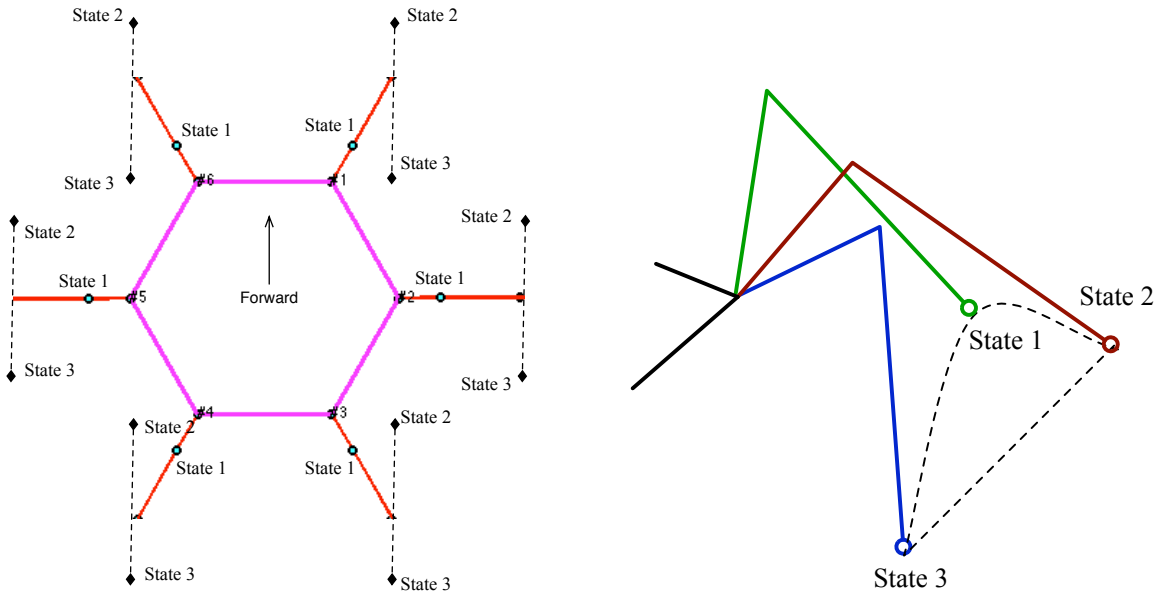
finite number of states should be defined in leg configurations. The robot explores the the state action domain and gets reward in every possible action. In every step, from every state of the agent, each action is rewarded and the rewards of specific state action is stored. Future actions are taken based on the actions done and specified rewards in a way which maximizes the coming reward in the present and future states and actions. The schematic of actions, states and rewards can be seen in figure 5.1.

It is desired here to learn the time sequencing of hexapod gait. The legs are moved in possible states with different specified actions. At the end of learning procedure, the sequence of actions in different states should be learned satisfying the goal which is walking on a straight line while minimizing tilt and other undesired translations.

5.2 State and Actions Specification in SiWaReL for Walking

Learning problem for six-legged walking robot requires defining some discrete state and actions because the large state and action number increases the computational costs exponentially. In this project, only three states are considered for each leg which mean 729 states for the robot. The first state is when the leg is not on the ground and lifted. The other two states are when the leg is on front and back of the stand or rest positions. Figure 5.2 shows the states of hexapod leg for walking problem.

$$\begin{aligned} S &= \left\{ \sum_{i=1}^{6} s_{j_i} 3^{(6-i)} | j = 1, 2..729 \right\} \\ &= \left\{ s_{j_1} s_{j_2} s_{j_3} s_{j_4} s_{j_5} s_{j_6 3} | s_{j_i} \epsilon \{0, 1, 2\}, i = 1..6 : i\text{th leg}, j = 1..729 \right\} \\ &= \{000000_3, 000001_3 ... 222222_3\} \end{aligned} \tag{5.1}$$



(a) States of legs in hexapod robot

(b) Defined states in each hexapod's leg

Figure 5.2: The defined states for reinforcement learning of walking in hexapod

$$s_j = s_{j_1} s_{j_2} s_{j_3} s_{j_4} s_{j_5} s_{j_6} 3 \tag{5.2}$$

For the goal of learning to walk, actions are defined as going to new state i.e. leg position. So each action can be specified by moving between the states. As in each state the robot can move to any other state. The actions space have the same dimension as the states which is 3^6.

$$A = \left\{ \sum_{i=1}^{6} a_{j_i} 3^{(6-i)} | j = 1, 2..729 \right\} \tag{5.3}$$

$$a_j = a_{j_1} a_{j_2} a_{j_3} a_{j_4} a_{j_5} a_{j_6} 3 \tag{5.4}$$

$$(s, a) \quad \in \quad S \times A \tag{5.5}$$

The learning procedure is a time consuming task due to the simulator which requires a certain time for dynamic simulation in each state. Also high number of iterations are needed for the robot to explore possible states actions set, update the rewards and find a desirable policy which results in the robot to walk. In early learning simulations, it took 10 days to update the Q matrix in Reinforcement learning to about 90 percent. In reinforcement learning, the robot explores the states and the actions with possible higher rewards. The state action is a 729^2 set. It is obvious that some parts of this set would not be explored at all or would definitely be with high punishment. For example, when the leg is on state 1 (lifted), moving to state 3 does not make sense or when it is on state 2, moving back to state 1 does not help the robot walk (considering positive cycling of walking). Hence, decreasing the state-action space by omitting the unreachable configurations or actions would help the computation speed of learning which is considerable.

A simulation has been executed to find all the possible actions for the robot to explore while considering omitting obvious impossible and non-sense actions such as the ones have been mentioned. In every state, due to leg configurations and positive leg cycling, possible actions are chosen. The simulation has reduced the set of state action space, and it is shown that only %8.64 (45927 out of 729^2) of all possible actions in different states are feasible for the robot to explore. Figure 5.3 shows these actions and states.

Inverse kinematic analysis requires a considerable computation time. Due to the fact that states and actions are defined specifically, the inverse kinematic of the states and actions are solved offline and in each state the joint variables are found and saved in table 5.1. In learning algorithm, in each state instead of solving inverse kinematic for the specific state, a table look up would be done.

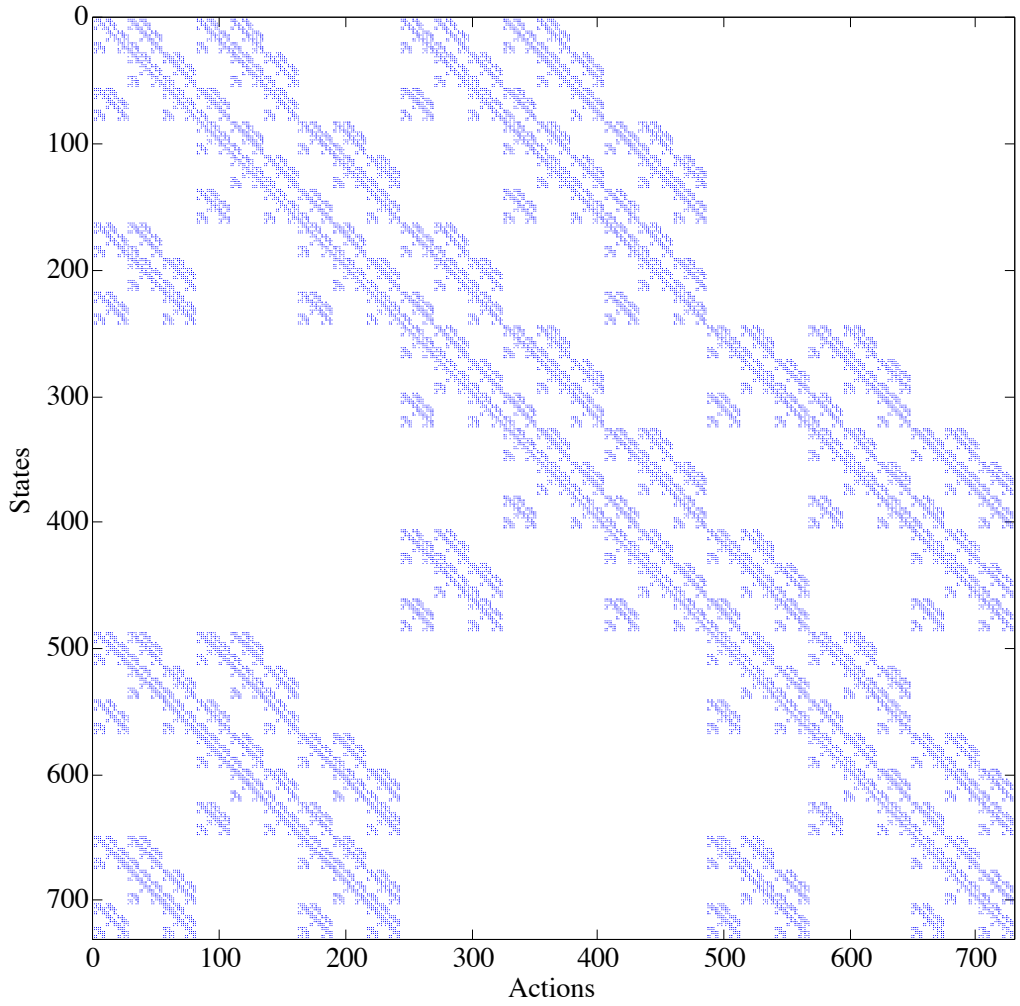


Figure 5.3: State action space with omitted non-sense actions regarding walking cycles

Table 5.1: The joint variables of six-legged walking robot in different defined states

Joint	state 1 (rad)	state 2 (rad)	state 3 (rad)
θ_{11}	0.5236	0.3370	1.0230
θ_{12}	-1.8928	-0.4593	-0.2759
θ_{13}	2.8055	1.8850	2.3446
θ_{21}	1.5708	1.0595	2.0821
θ_{22}	-1.8928	-0.4718	-0.4718
θ_{23}	2.8055	2.1230	2.1230
θ_{31}	2.6180	2.1186	2.8046
θ_{32}	-1.8928	-0.2759	-0.4593
θ_{33}	2.8055	2.3446	1.8850
θ_{41}	-2.6180	-2.1186	-2.8046
θ_{42}	-1.8928	-0.2759	-0.4593
θ_{43}	2.8055	2.3446	1.8850
θ_{51}	-1.5708	-1.0595	-2.0821
θ_{52}	-1.8928	-0.4718	-0.4718
θ_{53}	2.8055	2.1230	2.1230
θ_{61}	-0.5236	-0.3370	-1.0230
θ_{63}	-1.8928	-0.4593	-0.2759
θ_{63}	2.8055	1.8850	2.3446

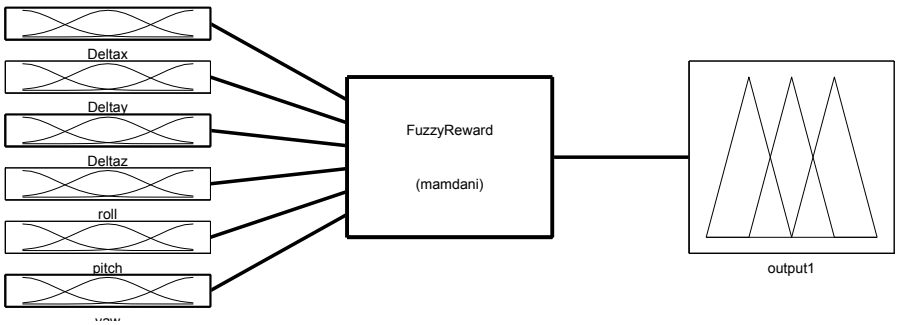


Figure 5.4: The diagram of fuzzy system which is used to generate reward values using sensors data (with respect to actions and states)

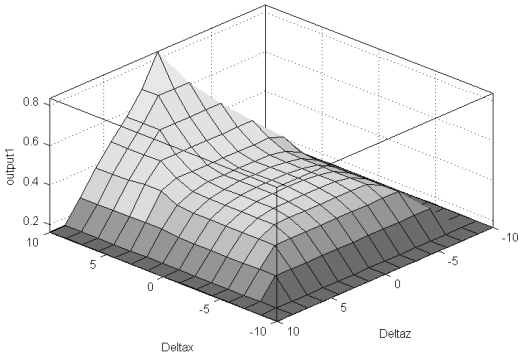
5.3 Fuzzy Reward

One of The challenges in this content is to establish the interaction between the environment and the six-legged robot. A system that tells the robot how good its movement or actions are. A typical way is using a mathematical function which uses sensory data and tells how good or bad an action is in a certain state. [28].

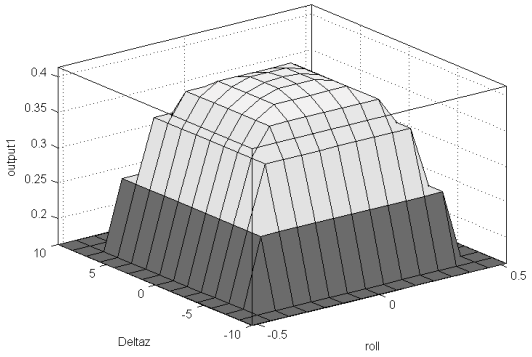
$$R(s,a) = \frac{forward}{\sqrt{translation^2 + tilt^2}} \tag{5.6}$$

Another approach which is presented and discussed here is a fuzzy inference system that uses sensory data and tells the reward value. The design of this system is shown in figure 5.4. Sensory data come from 3-axis digital compass, 3-axis digitall gyroscope and 3-axis digital accelerometer. Using these three sensors displacement and tilting can be found approximately using two integrators. 6 inputs to the Fuzzy Inference Systm (FIS) are displacements in x, y, and z direction and tilt around x, y, and z which are Δ_x, Δ_y, Δ_z, θ_x, θ_y and θ_z respectively.

It is assumed that moving in x direction is forward movement and is the desired movement and the other 5 inputs denote the parameters in other directions and movements which are undesired variables. So the surfaces of FIS are shown in figure 5.5. The surfaces of Δ_x to other 3 variables also are the same because of the definition of



(a) Surface of fuzzy output with respect to Δ_x and Δ_z



(b) Surface of fuzzy output with respect to Δ_z and θ_x

Figure 5.5: The surface of fuzzy system with respect to different inputs. Due to inputs characteristics similarity excluding Δ_x which is shown in (b) for example, the other surfaces will be the same as these two surfaces.

inputs. The simulations compare the defined FIS reward and mathematical reward in different states which are shown in figure 5.6. It should be mentioned that an offset and gain are applied to fuzzy reward signal for better comparison understanding. As the results show, the fuzzy reward makes more sense and has more useful information compared to mathematical reward function. However, the average calculation time is measured, and it is seen that fuzzy reward requires more time for computation. The average evaluation time for fuzzy reward in simulation is around 2 msec, but for mathematical one is 50 μsec. Evaluation time is so important in real-time control.

There are some approaches which reduce the number of calculations and complexity of fuzzy systems. One efficient way is to reduce the number of inputs which have similar properties or have been controlled with similar rules. A new combined variable with similar rule base is defined as (25) to reduce the number of calculations.

$$\Lambda = \sqrt{\Delta_y^2 + \Delta_z^2 + \theta_x^2 + \theta_y^2 + \theta_z^2} \tag{5.7}$$

where Λ denotes the tilt and translation. Therefore, Λ FIS with Δ_x and Λ as 2 inputs

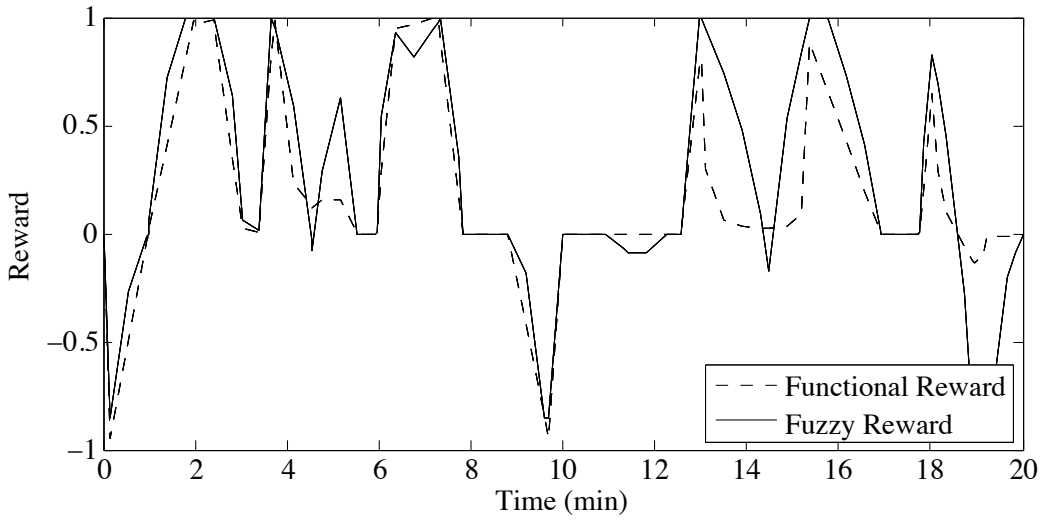


Figure 5.6: Signals of both functional reward which is mathematical formula and fuzzy reward. As shown, fuzzy reward signal is more detailed and accurate.

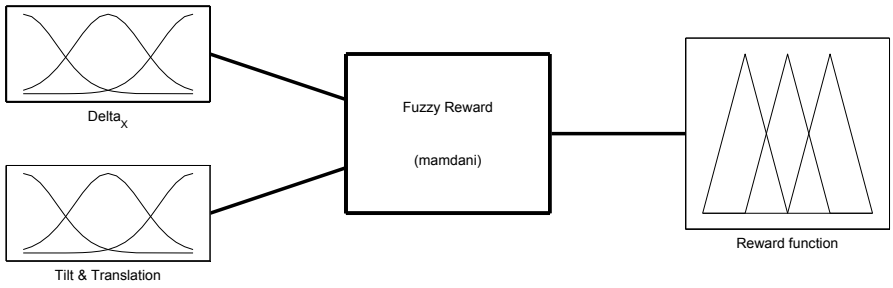


Figure 5.7: The diagram of Λ fuzzy reward

and reward as output can be defined. The new FIS structure is shown in 5.7. In addition, the rules surface is defined as before, but with new variables, the rules are decreased and the rules surface of the λ FIS is shown in figure 5.8.

The comparison shown in the figure 5.9 denotes that the efficiency of the FIS decreased slightly but compared to the mathematical reward function Λ FIS is more accurate. Also the evaluation time decreased to less than half of single evaluation in 6-input FIS. Mathematical reward is not as accurate as FIS systems, but, it evaluates the data at a higher speed. Time is a big deal in real time control and online processing. Fuzzy reward gives more accurate and detailed reward data. Although it

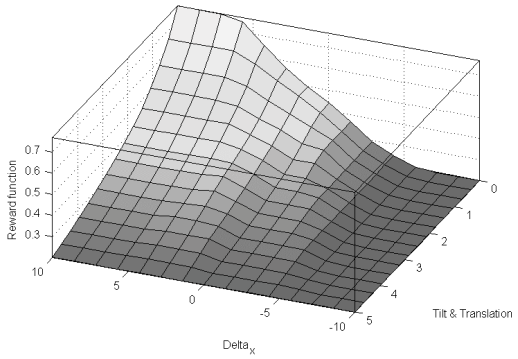


Figure 5.8: The surface of Λ fuzzy reward with respect to Δ_x & Λ.

is obvious that aggregation of inputs can speed up the FIS, it should be assured that there would not be much loss of data during input aggregation. In this case, as it is shown, data loss does not happen and the Λ Fuzzy reward works as accurate as the normal 6-input Fuzzy reward design but at higher speed.

Table 5.2: Speed performance comparison of different Rewarding systems

Reward	Time required for 1000 evaluations	Average time for single evaluation
Functional Reward	0.043 *sec*	0.043 *ms*
Fuzzy Reward	1.834 *sec*	1.834 *ms*
Fuzzy Reward with Λ	0.072 *sec*	0.72 *ms*

In Reinforcement Learning for walking, the reward signal is generated using Fuzzy Inference System and the comparison of the results with mathematical reward has shown that fuzzy reward gives more accurate data with slower computation compared to mathematical one. A new approach for reducing computation time in fuzzy system is applied and results have shown that the presented system works faster with the same accuracy. Therefore, the performance of the fuzzy reward has been improved.

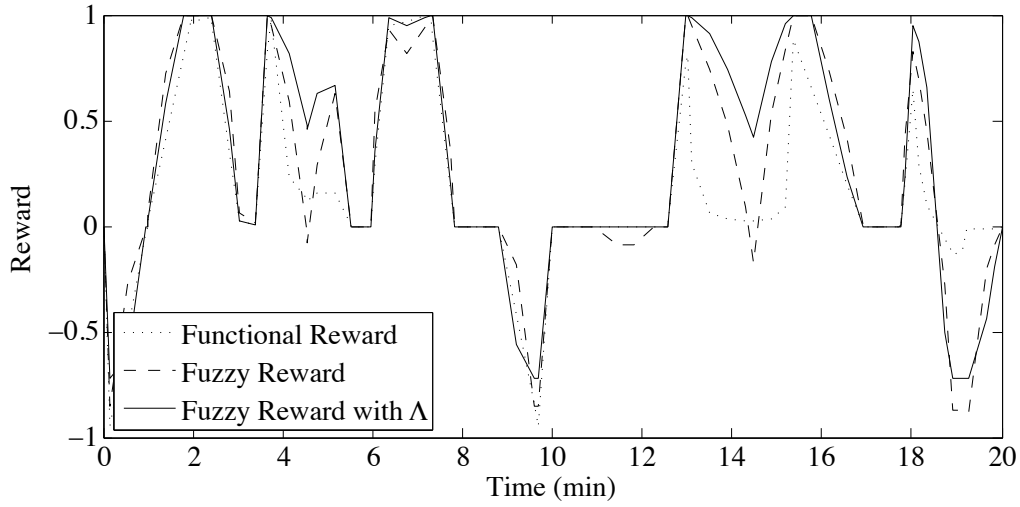


Figure 5.9: Comparison of reward evaluation of Λ fuzzy reward, the normal fuzzy reward and functional reward. It is shown that Λ fuzzy reward is slightly less accurate than the normal fuzzy reward.

5.4 Action Selection

A basic observation says that the action should be chosen with the best (highest) estimated payoff or in other words *greedy* choice. The simplest method is to choose the action with which $Q_a^* = \underset{a}{max} Q(a)$. However, the knowledge with this method would be exploited to actions with immediate highest rewards. So the learner who always chooses the greedy choice is not a good learner, because there would be a loss of explored places that it did not explore and failed to find the best action. There is another randomly action selection method called ϵ-greedy in which the learner chooses the actions randomly but with fixed $\epsilon > 0$ probability of the greedy choice. So as a good learner, it does not look for optimal action in every learning step, but it takes sub-optimal actions and saying, in other words it explores. In $\epsilon - greedy$ method using a fixed ϵ, an action is chosen randomly with a probability of ϵ; otherwise, the learner will go with the greedy action.

As it has been said, the actions can be taken randomly or by some action selec-

tion algorithms. Choosing the actions completely arbitrary is simple, not intelligent, results in poor performance and requires lots of time to explore every place of $S \times A$. There are other different kinds of action selection like Softmax action selection or "Boltzmann exploration" strategy in which a distribution of actions is defined by equation 5.8:

$$\pi(a) = \frac{e^{\beta Q_t(a)}}{\sum_{b=1}^{n} e^{\beta Q_t(b)}} \tag{5.8}$$

this is the controlled method of greedy choice where $\beta > 0$ controls the greediness of action choosing. When $\beta \to \infty$, it becomes greedy action selection [31]. So the greediness of the action depends on the specific possible actions in different part of learning. It is discussed in various researches that action selection is a problem dependent decision, and it can not be said that which action selection has better performance than the other. Both $\epsilon - greedy$ and Softmax action selection have their own benefits and drawbacks. For example, in Softmax β is a key factor in action selection and should be assigned carefully and also the same is for ϵ.

There are other action selection methods with better approaches that are good to be mentioned here like optimism in the face of uncertainty (OFU) principle [32] in which the learner chooses the actions with the highest upper confidence bound (UCB). A recent successful algorithm implements UCB with action a and time t in equation 5.9.

$$U_t(a) = r_t(a) + R\sqrt{\frac{2\log t}{n_t(a)}} \tag{5.9}$$

where n_t, r_t are the times that action a is selected and tried till time t and sample mean of rewards of n_t for a in $[-R, +R]$ respectively. It is shown that the failure probability of this algorithm is t^{-4}. If an action has not been tried, it increases its UCB value. [33].

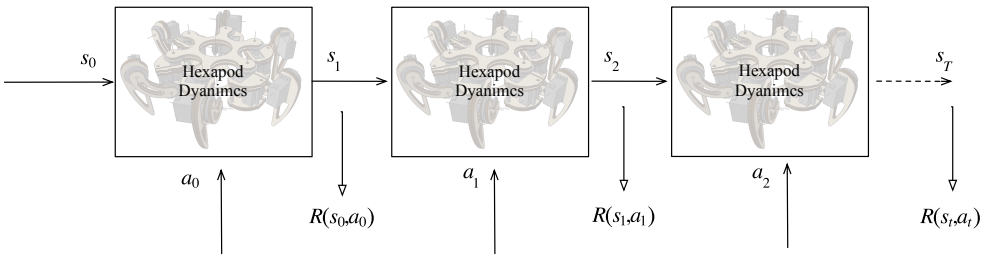


Figure 5.10: Policy search for hexapod walking learning

5.5 Learning algorithm

Although at the first step the aim was to implement the learning on the real built prototype of SiWaRel hexapod, the literature has shown that online learning requires a long time and in most of the same projects, circuits or motor burning was so common. Therefore, a dynamic model of the robot in simulation environment as the simulator is developed here to implement the reinforcement learning on SiWaRel hexapod robot and generate a free walking gait. The simulator task is to simulate the robot in dynamic environment as accurate as how the robot would act in the real world. In different states, different actions are taken and the reward is then calculated as it is illustrated in figure 5.10.

The robot starts from a state. It takes an action and goes to the next state, meanwhile the reward of the current state and action is calculated and stored. The simulator due to the taken action goes to the next state. An action is taken, the robot goes to the new state and the reward of the new action and state is calculated and this process goes on. The reward values should be stored in a way to be used in action selection and finding the best optimal policy. 5.10 shows how Q values are updated using s, a and $\Re$.

$$Q : S \times A \Longrightarrow \Re \tag{5.10}$$

Q-learning is a kind of reinforcement learning in which no model of the environ-

ment is needed. In Q-learning all the values are stored in a Q matrix. The simple Q-learning updates Q in every state and action by the immediate reward and the maximum Q value in the next state. As it is clear, the Q matrix should have the dimension of $s \times a$ which is 729^2.

$$Q(s,a) = R(s,a) + \lambda \max_a Q(s',a') \tag{5.11}$$

s', a' denotes the future action and state that would be taken. R and $\lambda < 1$ are the reward value and the discount factor respectively. It is shown that [28] the Kalman filtered version of Q-learning has better performance in this purpose as Burton and Sutton shown in text [30].

$$Q(s,a) = \alpha[R(s,a) + \lambda \max_a Q(s',a')] + (1-\alpha)Q(s,a) \tag{5.12}$$

So the learning starts from s_0 which all the legs are lifted. The chosen state is looked up in the state/inverse kinematic table and the joint values are fed into the dynamic robot simulator. An action, i.e. new state, is taken by the defined action selection policy and again the new joint variables are looked up in the state/inverse kinematic table and then are sent to the dynamic hexapod simulator. Then the simulation runs and the movement of the robot from the state to new state is simulated and the results of simulation which is movement and tilt in and around x, y and z are calculated. After finding out the simulator's result, then the fuzzy reward is calculated. The Q matrix is updated using the immediate reward, action, state and this procedure is repeated for the new states until the learning has been completed. As it is mentioned in the previous sections, a table look up for states and actions is used instead of inverse kinematic computation. Figure 5.11 shows how this loop works and Q is updated in every iteration

As it is said before, action in step t a_t is defined the new state s_{t+1}, but it should be mentioned that the opposite is not always correct. It means that s_{t+1} is not always

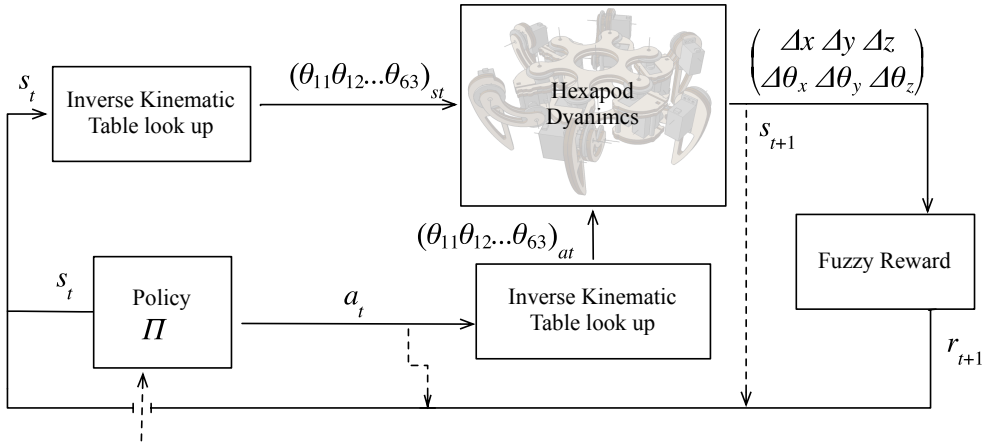


Figure 5.11: The schematic of reinforcement learning for hexapod robot

a_t. Assuming a time that a leg has became malfunctioned or is not working properly, in this situation a_t which is defined as new state is sent to the simulator and although the robot tries to go to new state, s_{t+1} is something different that a_t tries to go. This is the fault tolerant feature of online learning which reinforcement learning provides. Even if a failure has been caused in one or two legs, it can learn how to walk with the other healthy legs. So the algorithm of learning can be summarized to algorithm 1.

The procedure of learning starts at initializing $s_0 = 000000$ and $Q = 0$. In each iteration, the action is decided by the defined action selection policy π. The join angles of the state and action are assigned from the state/inverse kinematic table 5.1. The learning iterations continue until the learning is completed. However, it can run forever and update as the robot is completing a certain task to help itself in failures and other possible problems that would happen.

Algorithm 1 Q-learning algorithm of the SiWaRel hexapod robot

procedure Q(s, a)
  $s \leftarrow s_0$
  initialize Q
  repeat
    $a \leftarrow \pi(s)$ ▷ ϵ-greedy action selection
    $\Theta_1 \leftarrow \Gamma(s)$ ▷ State/inverse kinemtaic Table
    $\Theta_2 \leftarrow \Gamma(a)$
    $\Delta, s' \leftarrow Sim(\Theta_1, \Theta_2)$ ▷ Execute dynamic hexapod simulator
    $\Re(s, a) := \Lambda(\Delta)$ ▷ Fuzzy reward
    $Q(s, a) := \alpha[\Re(s, a) + \lambda \max_a Q(s', a')] + (1 - \alpha)Q(s, a)$
    $s \leftarrow s'$
    $n := n + 1$
  until $|Q| < q_{max}$
  return Q ▷ The optimum policy
end procedure

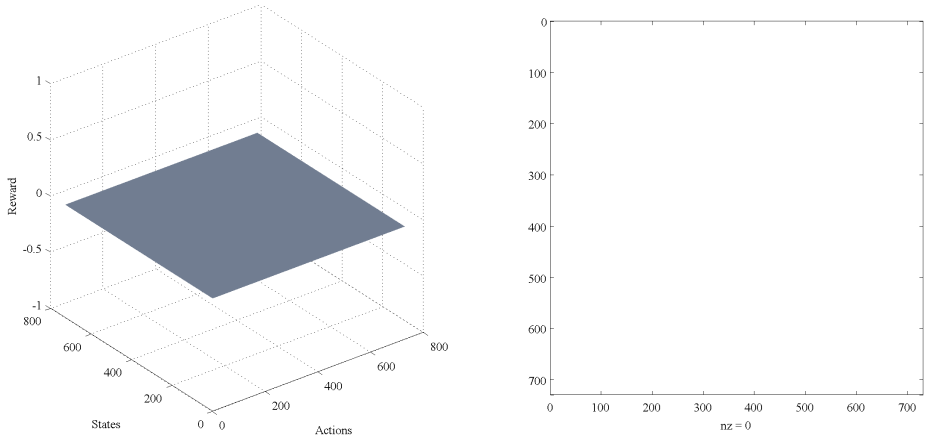
5.6 Results

Using the algorithm defined in the previous section and dynamic simulator developed in chapter 4, the procedure of learning has been executed for a big number of iterations.

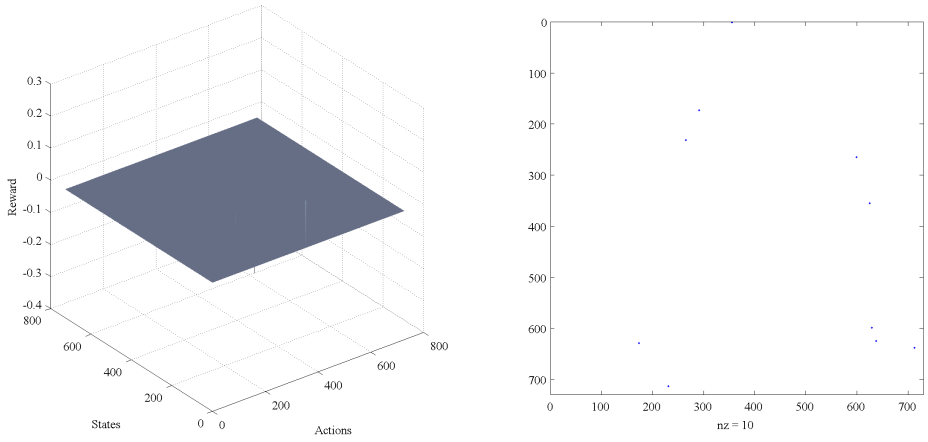
The learning procedure requires a considerable time. As in each simulation, the dynamic behavior of hexapod is needed to simulate the sensory measurements and state transitions as accurately as it would happen in the real world. Each simulation of every state action takes about 2.5 seconds excluding the initializing, storing and transferring data time. Also the iteration itself has its own certain required computation time including fuzzy reward calculation. Executing the MATLAB Simulink files in every learning iteration is also a time consuming procedure despite the fact that the simulation itself requires its own simulation time from compiling, initializing, running, storing the results and sending back to MATLAB work space for the rest computation of learning iteration.

5.6.1 Random Action Selection

The early learning procedure took around 10 days of simulation and executing learning loop to update Q to about %98. Figures 5.12 to 5.14 show the trend of learning and updating Q using randomly chosen actions approach.

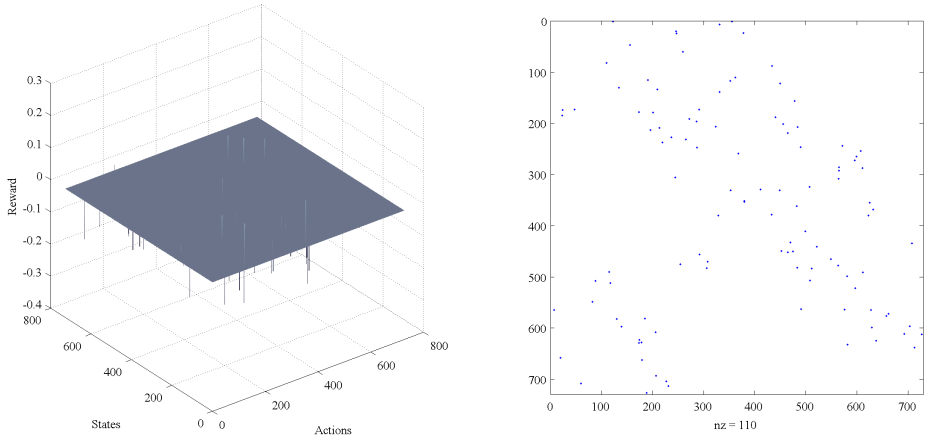


(a) Q matrix in starting of learning iterations

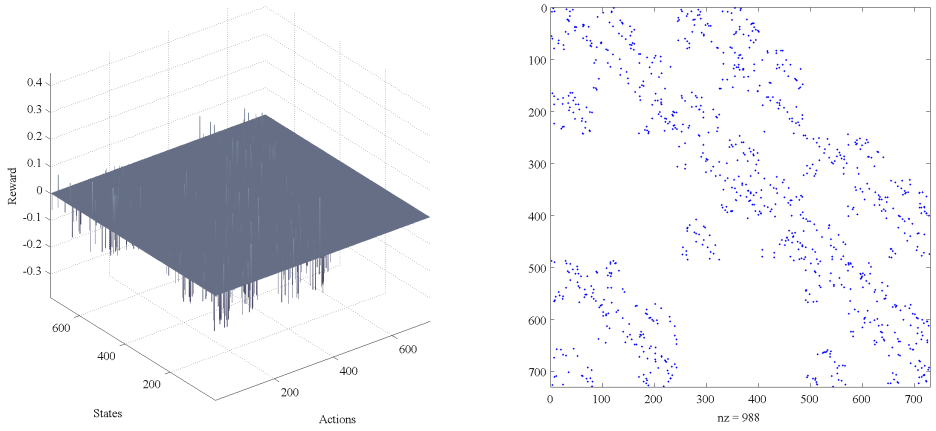


(b) Updated Q matrix in 10 learning iterations

Figure 5.12: The progress of SiWaRel learning to walk iterations



(a) Updated Q matrix in 100 learning iterations



(b) Updated Q matrix in 1000 learning iterations

Figure 5.13: The progress of SiWaRel learning to walk iterations

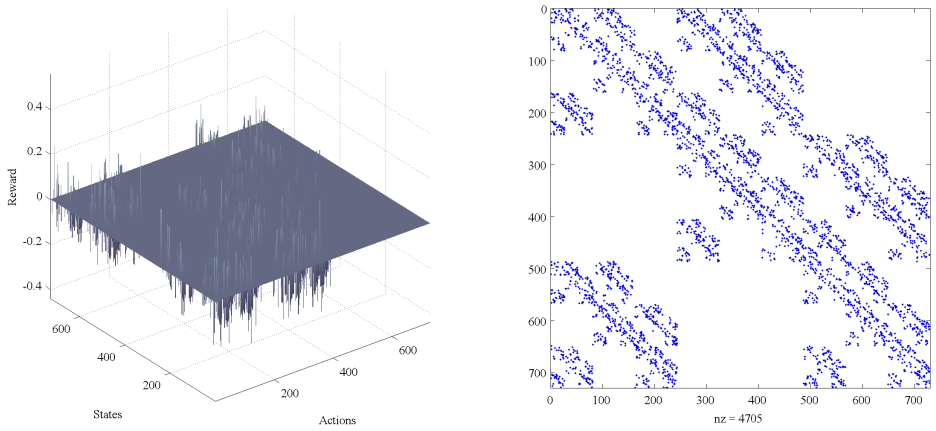


Figure 5.14: Updated Q matrix in 10000 learning iterations

5.6.2 ϵ-greedy Action Selection

ϵ-greedy is a sub optimal action selection which, as the literature shows, is promised to have a better performance due to the fact that a good learner does not always choose the optimal choice. Therefore, by changing the greediness of action selection, it is shown that Q matrix is updated with spending less time on exploring non optimal places in state action space. The results of updating Q with $\epsilon = 0.1$, $\epsilon = 0.2$ and $\epsilon = 0.5$ are shown in figure 5.15 and 5.16.

All come to updated Q with approximately acceptable tolerance. However as it can be seen bigger ϵ (i.e. less greediness) explores wider state-actions space but requires bigger number of iterations to find the optimal actions.

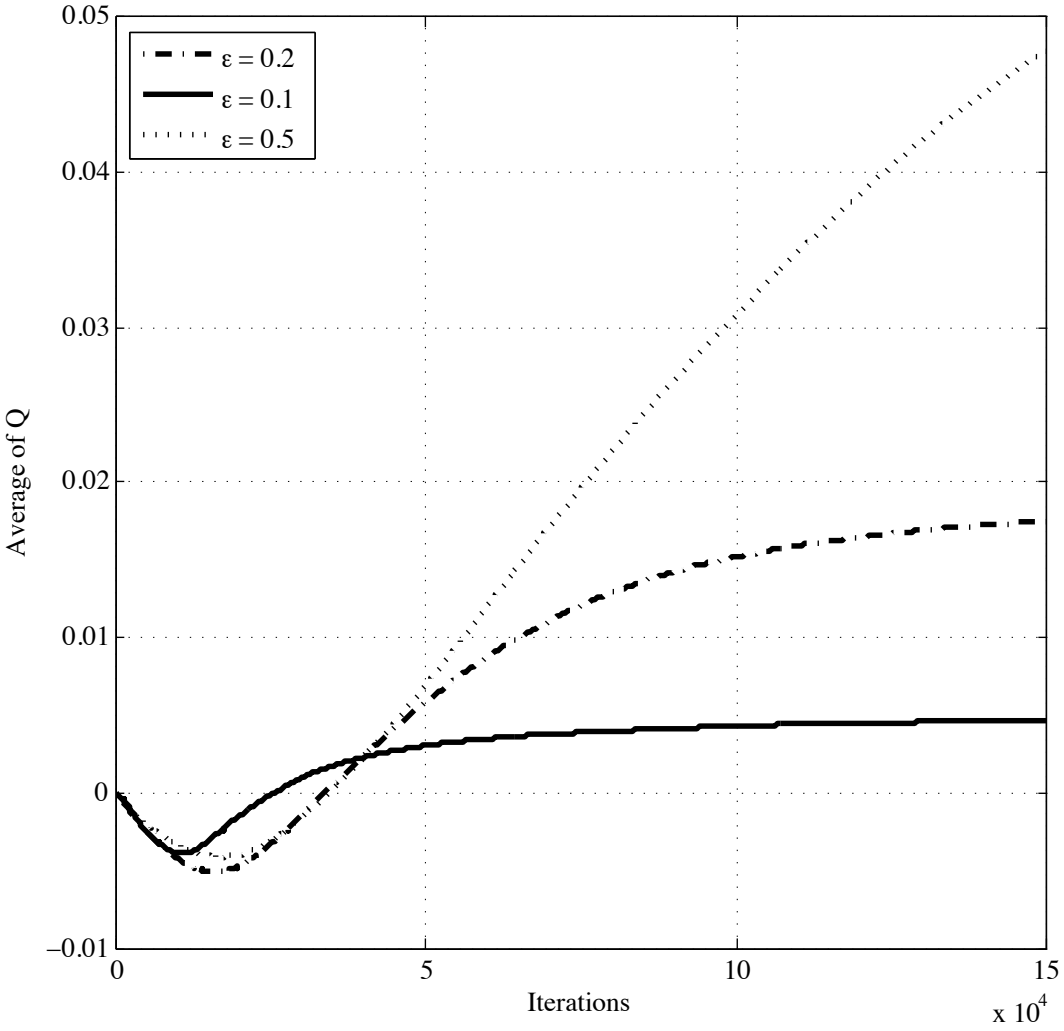
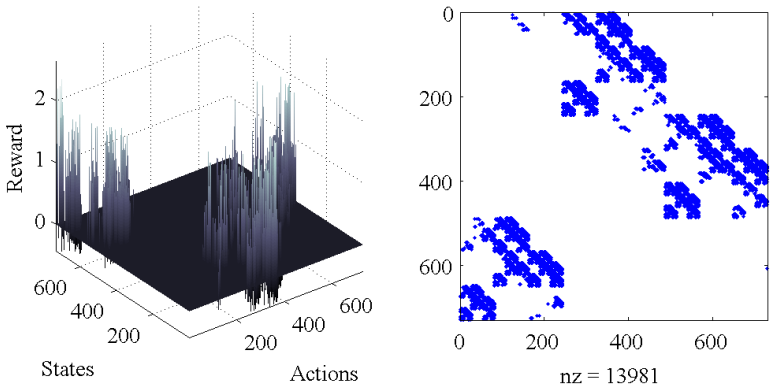
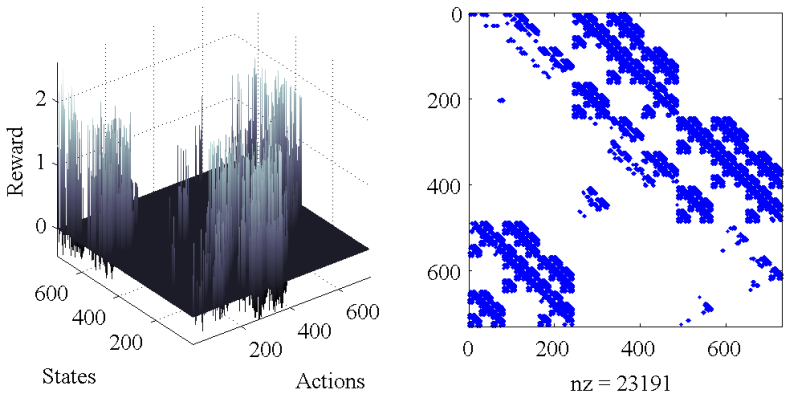


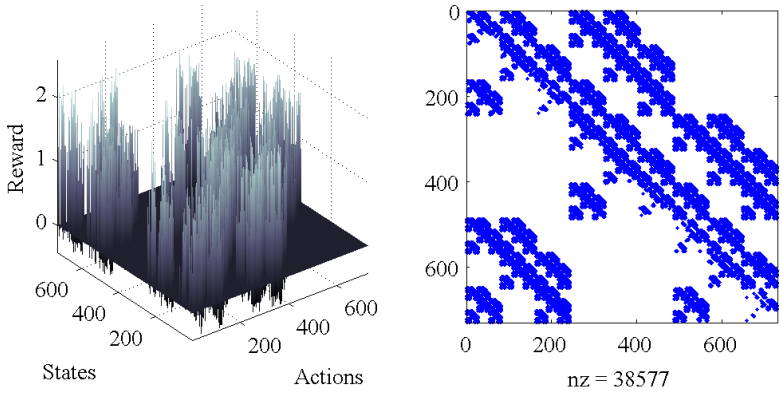
Figure 5.15: Updating trend of Q matrix to 150000 learning iterations with ϵ-greedy action selection, $\epsilon = 0.1, 0.2$ & 0.5



(a) $\epsilon = 0.1$.



(b) $\epsilon = 0.2$.



(c) $\epsilon = 0.5$.

Figure 5.16: Updated Q after 150000 Iterations with different greediness. The right figures show the explored state action space.

5.6.3 Decision making

The updated Q at the end of the learning procedure defines the best policy which tells what action $a = \pi^*(s)$ to be taken in different states.

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} Q(s, a) \tag{5.13}$$

so in each state the optimal action will find out using 5.13. This means the robot decides the next action by the experience it had during the learning procedure. As it is shown, the Q depends on immediate reward and the upcoming reward in possible future states.

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots \tag{5.14}$$

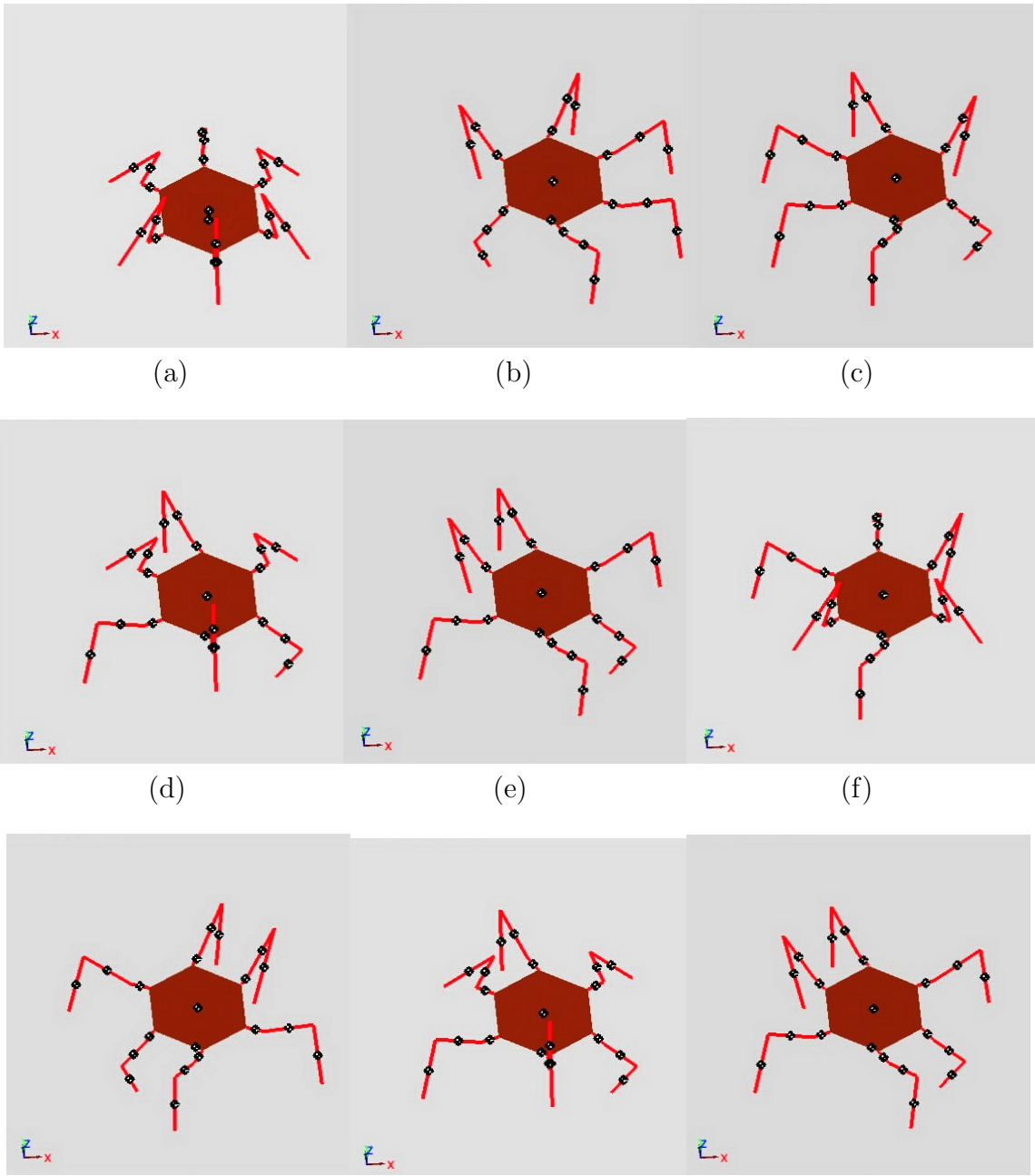
$$\text{Total Payoff} : R(s_0, a_0) + \lambda R(s_1, a_1) + \lambda^2 R(s_2, a_2) + \lambda^3 R(s_3, a_3) + \dots \tag{5.15}$$

Therefore, the optimal policy results in best actions with the highest pay off.

Implementing the learning results on the hexapod dynamic model shows that the hexapod robot learned to walk well using the fuzzy system for rewarding in reinforcement learning. As it is shown in figure 5.17 the robot is walking in the right direction using the optimal policy.

5.7 Summary

The learning procedure of SiWaRel hexapod robot has been discussed in this section. The states and actions for learning problem is defined in some discrete states and actions and it is shown that only a small part of state action state is explored during learning and walking. The rewarding function for learning is discussed and a fuzzy approach is developed. The results have shown that the fuzzy reward has better performance compared to other reward funtions have been used in other similar works.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5.17: Evaluating the optimal policy on the dynamic model of hexapod robot. As it can be seen, the robot is moving on the desired direction ($+x$).

At the final steps, the Q-learning approach is used to find the best policy of action taking for the hexapod robot to learn to walk. It is shown that this approach has benefits that if a failure happens to the legs of the robot, it can learn to walk with the malfunctioned legs, as it would get the best reward and total pay off with its situation anyway.

Chapter 6

Conclusion

An 18 DoF hexapod walking robot has been designed, built, and studied in this thesis. The kinematic analysis of the robot has been formulated using a modular view approach considering 6 DoF for the robot trunk. Two typical gaits have also been studied using the presented kinematic formulation and all the analysis has been verified in both simulation and implementation on experimental prototype.

Free gait generation of hexapod walking robot is studied with a reinforcement learning approach. For the goal of finding the rewards in state action space a dynamic simulator of the hexapod walking robot has been developed using objective oriented programming. A fuzzy inference system is designed for generating reward signal in different states-actions, and it is shown that the fuzzy reward approach generates more accurate rewarding signal than other kind of rewards discussed in the literature. The learning of walking then has been completed after days of simulation using Q-learning which is also a type of reinforcement learning, and it is shown that this online learning has advantages such as fault tolerancy and capability of learning to walk on different terrains.

Bibliography

[1] C. Prahacs, A. Saudners, M. K. Smith, D. McMordie, and M. Buehler, "Towards legged amphibious mobile robotics," Proceedings of the Canadian Engineering Education Association (2011).

[2] "Wikipedia, Six-legged Walking Robot,".

[3] A. Mahajan and F. Figueroa, "Four-legged intelligent mobile autonomous robot," Robotics and Computer-Integrated Manufacturing **13,** 51–61 (1997).

[4] F. Hardarson, "Locomotion for difficult terrain," Dept. Mach. Des., Royal Inst. Technol., Stockholm, Sweden, Tech. Rep. TRITA-MMK **3,** 1400–1179 (1998).

[5] Y. Ota, Y. Inagaki, K. Yoneda, and S. Hirose, "Research on a six-legged walking robot with parallel mechanism," In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, **1,** 241–248 vol.1 (1998).

[6] P. González de Santos *et al.*, "Using walking robots for humanitarian de-mining tasks," (2004).

[7] B. S. Lin and S.-M. Song, "Dynamic modeling, stability, and energy efficiency of a quadrupedal walking machine," Journal of Robotic Systems **18,** 657–670 (2001).

[8] J. E. Clark, J. G. Cham, S. A. Bailey, E. M. Froehlich, P. K. Nahata, R. J. Full, and M. R. Cutkosky, "Biomimetic design and fabrication of a hexapedal running robot," In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, **4,** 3643–3649 (2001).

[9] B. Siciliano and O. Khatib, *Springer handbook of robotics* (Springer, 2008).

[10] J. E. Clark, Ph.D. thesis, stanford university, 2004.

[11] T.-T. Lee, C.-M. Liao, and T.-K. Chen, "On the stability properties of hexapod tripod gait," Robotics and Automation, IEEE Journal of **4,** 427–434 (1988).

[12] P. G. de Santos, E. Garcia, and J. Estremera, "Quadrupedal locomotion,", 2006.

[13] K. Waldron and R. McGhee, "The adaptive suspension vehicle," Control Systems Magazine, IEEE **6,** 7–12 (1986).

[14] R. D. Beer, R. D. Quinn, H. J. Chiel, and R. E. Ritzmann, "Biologically inspired approaches to robotics: what can we learn from insects?," Communications of the ACM **40,** 30–38 (1997).

[15] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A simple and highly mobile hexapod robot," The International Journal of Robotics Research **20,** 616–631 (2001).

[16] "Prototype This! build a full scale Rhex,", http://videos.howstuffworks.com/discovery/34376-prototype-this-six-legged-all-terrain-vehicle.

[17] "WebX hexapod project,", http://www.webx.dk/robot-crawler/robot-crawler.htm.

[18] P. Younse and H. Aghazarian, "Steerable hopping six-legged robot," In *SPIE Defense and Security Symposium*, pp. 69600H–69600H (2008).

[19] Pulstech, "Timberjack, december 2009,", http://www.plustech.com.

[20] S.-M. Song and K. J. Waldron, "An analytical approach for gait study and its applications on wave gaits," The International Journal of Robotics Research **6,** 60–71 (1987).

[21] C.-D. Zhang and S.-M. Song, "Stability analysis of wave-crab gaits of a quadruped," Journal of Robotic Systems **7,** 243–276 (1990).

[22] S.-M. Song and K. J. Waldron, *Machines that walk: the adaptive suspension vehicle* (The MIT Press, 1989).

[23] K. Lilly and D. Orin, "Efficient dynamic simulation for multiple chain robotics mechanisms," In *Proceedings of 3rd annual conference aerospace computational control, Pasadena*, pp. 73–87 (1989).

[24] Z. Wang, X. Ding, A. Rovetta, and A. Giusti, "Mobility analysis of the typical gait of a radial symmetrical six-legged robot," Mechatronics **21,** 1133–1146 (2011).

[25] S. Shah, S. Saha, and J. Dutt, "Modular framework for dynamic modeling and analyses of legged robots," Mechanism and Machine Theory **49,** 234–255 (2012).

[26] M. García-López, E. Gorrostieta-Hurtado, E. Vargas-Soto, J. Ramos-Arreguín, A. Sotomayor-Olmedo, and J. Morales, "Kinematic analysis for trajectory generation in one leg of a hexapod robot," Procedia Technology **3,** 342–350 (2012).

[27] G. Figliolini, S. Stan, and P. Rea, "Motion analysis of the leg tip of a six-legged walking robot," In *Proceedings of the 12th IFToMM World Congress, Besançon, France*, (2007).

[28] M. R. Bunting and J. Rogers, "Q-Learning Hexapod (May 2009)," .

[29] S. B. Thrun, "Efficient exploration in reinforcement learning," (1992).

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (Cambridge Univ Press, 1998), No. 1.

[31] C. Szepesvári, "Algorithms for reinforcement learning," Synthesis Lectures on Artificial Intelligence and Machine Learning **4,** 1–103 (2010).

[32] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," Advances in applied mathematics **6,** 4–22 (1985).

[33] T. Jaksch, R. Ortner, and P. Auer, "Near-optimal regret bounds for reinforcement learning," The Journal of Machine Learning Research **99,** 1563–1600 (2010).