DESIGN AND DEVELOPMENT OF A WALKING APPARATUS

A project report submitted

in partial fulfillment of the requirement for the degree of

Bachelor of Technology

By

Lerrel Joseph Pinto

Roll No: 10010337

Shreeyash Uday Lalit

Roll No: 10010364

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

April 2014

APPROVAL SHEET

The project report entitled **'Design and Development of a Walking Apparatus'** by Mr. Lerrel Joseph Pinto (10010337) and Mr. Shreeyash Uday Lalit (10010364) is approved for the degree of Bachelor of Technology, Mechanical Engineering.

Examiner(s)

_____

_____

_____

Supervisor(s)

_____

_____

Chairman

_____

Date:

Place:

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand

that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Lerrel Joseph Pinto (10010337)
Department of Mechanical Engineering,
Indian Institute of Technology - Guwahati

Shreeyash Uday Lalit (10010364)
Department of Mechanical Engineering,
Indian Institute of Technology - Guwahati

CERTIFICATE

The work contained in this thesis titled **'Design and Development of a Walking Apparatus'** by Mr. Lerrel Joseph Pinto and Mr. Shreeyash Uday Lalit, both students of the Department of Mechanical Engineering, Indian Institute of Technology Guwahati has been carried out under my supervision. This work has not been submitted elsewhere for the award of any degree.

Dr. S K Dwivedy,

Professor, Department of Mechanical Engineering,

Indian Institute of Technology – Guwahati

Dr. Harshal B. Nemade,

Professor, Department of Electronics and Electrical Engineering,

Indian Institute of Technology – Guwahati

ACKNOWLEDGEMENTS

We would like to sincerely thank our advisors Dr. S K Dwivedy and Dr. H B Nemade for their constant support and guidance during the project. We would also like to thank Mr. Manouranjan Duarah, Mr. Krishanu Roy, Mr. Bhaben Kalita and other assistant staff members in the Mechatronics Laboratory for their logistic support. We further thank Mr. Dhiraj Gandhi and Mr. Sriram Kumar for their help in prototyping.

And lastly, we are deeply indebted to the Mechanical department of Indian Institute of Technology Guwahati for providing us with the opportunity – means as well as methods – for carrying out this work.

Lerrel Joseph Pinto (10010337)
Department of Mechanical Engineering,
Indian Institute of Technology - Guwahati

Shreeyash Uday Lalit (10010364)
Department of Mechanical Engineering,
Indian Institute of Technology - Guwahati

ABSTRACT

Since the past 30 years, significant research has gone into emulating the natural gaits that are observable in nature such as those seen in animals that display quadrupedalism. Quadrupedalism refers to a form of terrestrial locomotion inspired from animals and reptiles that walk with four limbs and the concept finds varied applications in everyday life ranging from usage in the armies as robotic mules to human transportation machines to relief and mitigation devices that might be able to intervene and enter radiation affected sites such as the recent horrific incident in Fukushima. The possibilities are enormous and the benefits unimaginable as quadrupeds have a clear advantage of traversing rugged terrain and obstacles in a much natural and suitable manner as compared to wheeled machines. Yet the practicality of implementing them in a real world scenario carries many obstacles such as energy expenditure, stability analysis in rugged harsh terrain and most importantly, autonomous motion planning through active sensor feedback and terrain detection. The major challenge in extending these procedures to a quadruped lies in the generation and computational expense of integration of the dynamic equations. The reason for this complexity is the high degree of freedom associated with a quadruped in comparison with a biped.

The first step is the literature review of the state of the art when it comes to humanoid joint and actuator designs which would give us an idea of the existing paradigms and robots which use various mechanisms for joint actuation. This phase would also include learning and familiarizing ourselves with the latest technology and equipment being used in the field upon which our work too will be invariably based. That would be supplemented by a preliminary comparison between a kneed walker and a quadruped to further understand the passivity of such systems.

Subsequently, the designing of a quadruped of 12 degrees-of-freedom would be initiated so as to implement paradigms of stability, efficiency and motion planning. It would first entail analyzing the dynamics and the gaits to be incorporated into the desired quadruped, followed by assimilating all the requisite materials to build the robot. Thereafter, different gaits would be implemented on the quadruped to check its performance along with incorporating different motion planning techniques.

TABLE OF FIGURES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Robotics is essentially the concept of creating machines that lessen human effort and can operate autonomously. The idea dates back to classical times, but research into its potential uses and applicability was not fully realized till the 20th century. The role of robotics has often been to mimic human behavior and manage tasks in a similar fashion. Today, this field is so rapidly growing that the research, design, and building of new robots serve various practical purposes, whether domestic, commercial or military. Another aspect that has been gaining momentum in recent years is that of walking machines and their extraordinary benefits and advantages over wheeled robots. However, the major obstacle that needs to be tackled in all these different applications is that of energy efficiency, stability and motion planning. The plan is to incorporate all these aspects into one model and find optimal configurations for walk-gait of a quadruped.

The following chapters will introduce the paradigm of 'Limit Cycle Walking' or more simply Passive Dynamic Walking. This concept has been previously used and applied in the design of two-legged machines with much success that has enabled them to perform to unprecedented levels with regards to speed, efficiency, disturbance rejection and versatility. Passive walking as a concept essentially puts fewer artificial constraints to the robot's walking motion allowing them to use their natural dynamics as compared to actuated robots whose gaits are unnatural and also very energy intensive. The artificial constraints are useful to address the complex problem of stabilizing the dynamic walking motion in robotic walking, whether it be in bipedal or quadrupedal robots. Artificial stability constraints invariably enable the creation of robotic gait while at the same time restrict the free-flowing natural gait and severely limit the performance that could be obtained. Therefore, as a ground rule, it is safe to say that there is less freedom for optimizing the performance if the constraints get more restrictive in nature.

A detailed understanding of passivity would then be followed by an attempt at providing a rudimentary understanding to the reader, pertaining to the basics of each subject that have been addressed, like Motion Planning, Control Architecture, etc. With respect to motion planning, all

algorithms pertaining to quadrupeds and their effective obstacle avoidance or path following aspects would be subsequently handled in the Literature Review section.

This section would then be concluded by the Objective of this research and an overview of the Organization of the report.

1.1 Static v/s Dynamic Walking

A brief classification is indicative of where this project will fit in. Firstly, there is a great need to distinguish between what are called static and dynamic machines. The first successful creation of bipedal robots in the early 70's was done through the oldest and most constrained paradigm which is, ‘static stability’.

Static stability primarily suggests that the Center of Mass vertical projection stays within the support polygon made by the feet. Their purpose is to maintain static equilibrium throughout their motion and majorly achieve this through a requirement of at least 4 legs, more commonly six. These forms of machines also impose a speed restriction, since it is important to limit cyclic accelerations in order to minimize inertial effects. Even though it is very simple to understand and apply, yet it leaves much room for improvement.

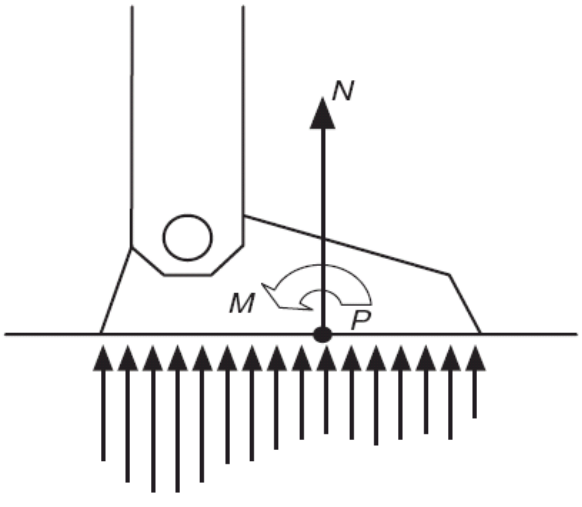


Figure 1.1 Typical description of foot forces in legged robots [3]

Therefore, most humanoid systems in the world presently use the more advanced ‘Zero Moment Point’ (ZMP) concept which was introduced in 1970 by Vukobratovic [1]. This ZMP concept dictates the stance foot to remain in flat contact with the surface at all times, therefore ensuring stability. Even though this constraint is less restrictive than static walking as the Centre of Mass can move beyond the boundaries of the support polygon, yet it still poses its own limitations with respect to efficiency and stability. Therefore, these robots are still under-achieving in those aspects as compared to human walking [2].

On the other hand, dynamic machines resemble people, in that they extend upon the natural motions with which most animals/mammals walk, which in turn can be causes them to have fewer legs than static machines and also makes them faster.

1.2 Walking Stability

In bipedals and quadrupeds, stability is not a simple concept [4, 5, 6]. Further to understand the walking robots completely, the subject of Limit Cycle Walkers needs to be addressed. For this purpose it is imperative to first classify the robots on the basis of walking stability.

To answer the basic question of stability, the most rudimentary understanding involves ‘to avoid falling’. This concept is reiterated with the ‘viability kernel' [7], where from a union of all feasible states, a stable state is derived based on which a walker is able to avoid a fall. This set of states will also involve all possible periodic movements or static states. The presence of possible disturbances should also be considered in the making of the viability kernel.

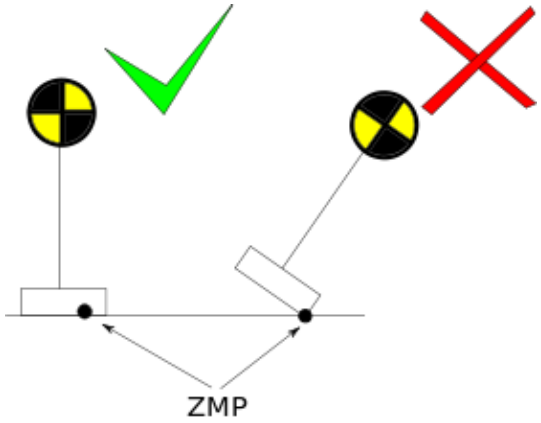


Figure 1.2 ZMP in legged robots [3]

Legged walkers should ultimately be designed using this idea of stability to optimize performance without any restrictions. However, because of the highly non-linear relationships with state space and gait configurations, it turns out that this method is not practical for gait synthesis. To establish stability using this definition requires a complete dynamic simulation or actual experiment beginning with all possible initial configurations of the walker, including all possible disturbances, checking whether the state leads to a fall or not. Since walking involves a lot of complex dynamics, this would be very expensive in both approaches - numerical as well as experimental.

A large group of robotic researchers have been led by the limited practical value of the concept of 'avoiding to fall' as a stability definition for gait synthesis, which has resulted them to [8, 9, 10] create a fairly restrictive distinction of stability for legged robots. This stability distinction is defined by them as 'sustained local stability'. Gait synthesis is performed using a desired trajectory through state space which is continuously implemented by incorporating stabilizing trajectory control. Even though this approach aims at sustained local stability, it can be proven by obtaining every point on the nominal trajectory that points in its local neighborhood in the state space converge to the trajectory.

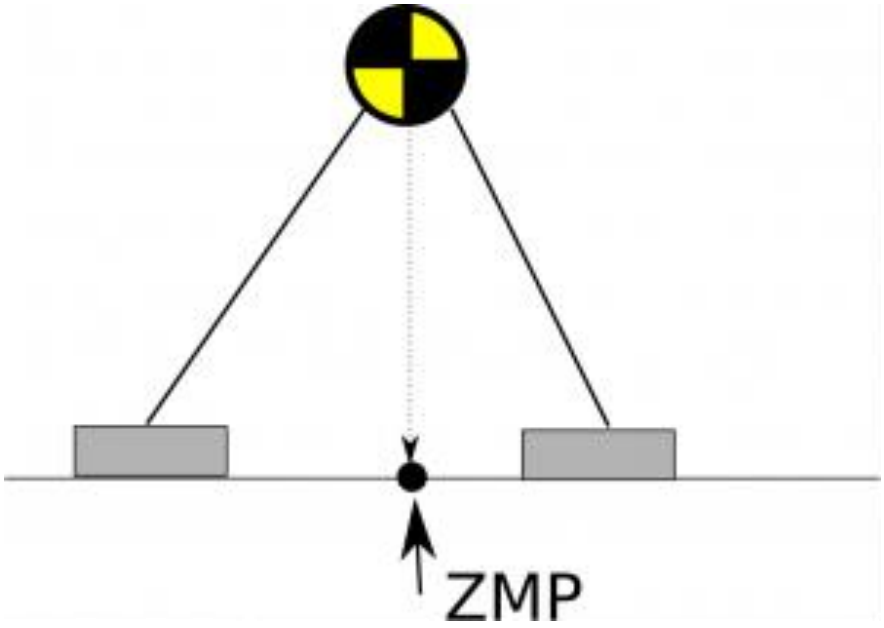


Figure 1.3 ZMP during walking [3]

Two important constraints need to be considered in achieving the aim of sustained local stability. Firstly it requires local stabilizability and secondly it needs high control stiffness. Local stabilizability exists only when at least one foot is firmly placed on the ground. This constraint is

however guaranteed by satisfying the Zero Moment Point (ZMP) or Center of Pressure (CoP) criterion [2, 4]. Further, the requirement to obtain local stability in the presence of the inherent unstable inverted pendulum dynamics of the single stance phase justifies the need of high control stiffness.

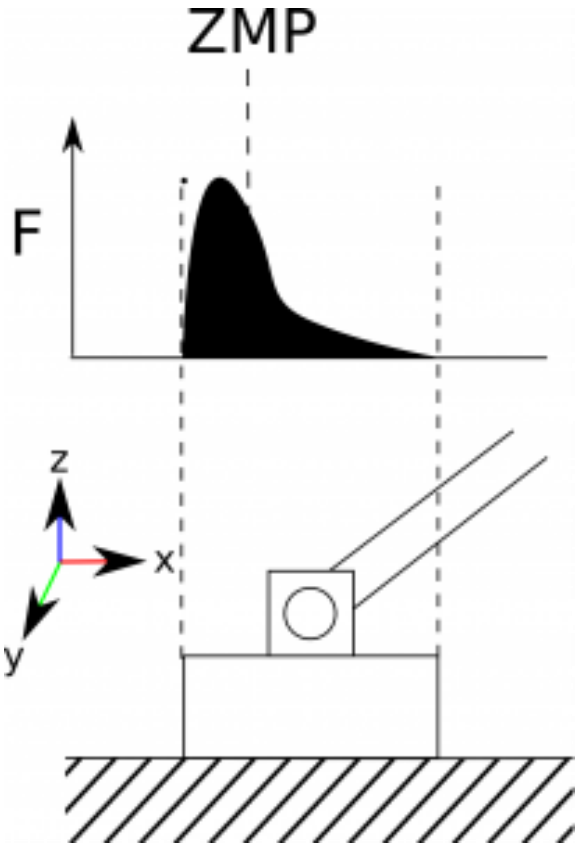


Figure 1.4 ZMP in legged robots [3]

The conventional definition of the ZMP is that it is a point on which the inertial and gravity moments cancel out and result in a net zero moment. This definition is usually not helpful and mostly used to confuse non roboticists. The simpler definition of a point of consideration is that it is simply the center of pressure (COP). The claim that the ZMP is equivalent to the COP led to much controversy initially but eventually the conceptually much more useful COP clearly came out on top. The center of pressure (COP) is the average of the pressure distribution it is defined mathematically as follows:

$$x_{zmp} = \frac{\int x * F_z(x)\, dx}{\int F_z(x)\, dx} \tag{1.1}$$

This can be calculated easily from a force-torque sensor by comparing the total downwards force vs the transverse torque. So for the x direction for example as follows:

$$x_{zmp} = \frac{\tau_y}{F_z} \tag{1.2}$$

The reason why COP/ZMP is important is that it allows one to gage the stability of the foot. For example, if the ZMP moves towards the edge of the foot, then the foot starts tipping over. Uncontrolled tipping is not good and often leads the robot to fall over.

1.3 What is Limit Cycle Walking?

To achieve an increase in the performance of bipedal and quadrupedal robots, a growing group of researchers have adopted a new paradigm for synthesizing gait and letting go of the restrictive aim for sustained local stability. This new paradigm is called 'Limit Cycle Walking'.

The definition of 'Limit Cycle Walking' is as follows: *Limit Cycle Walking is essentially a periodic sequence of steps that is stable as over the whole period but not locally stable at every instant in time.*

Mathematically speaking, in the study of dynamic systems with a two dimensional phase space, the limit cycle is a closed trajectory in the phase space that has the property that at least one other trajectory spirals into the closed trajectory either as time approaches infinity or when time approaches negative infinity. This behavior is commonly exhibited in several nonlinear systems.

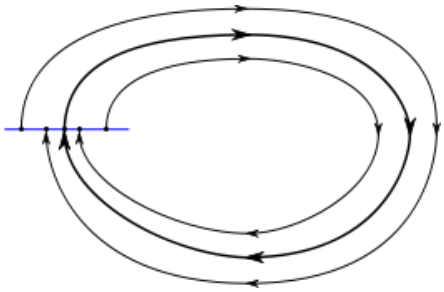


Figure 1.5 Limit Cycle in state space [3]

In the definition of Limit Cycle Walking, the periodic sequence of steps refers to the intended walking motion which is a series of exact repetitions of a closed trajectory in state space (a limit cycle), while placing the walker's feet forward: two in turn for a bipedal walker and four for a

quadrupedal walker. This definition implies that a stable trajectory may not locally stable at every instant in time. By this the need to make all points on the trajectory attract to their local neighborhood in state space as done in conventional control is negated. However, the motion is considered to be stable because eventually, neighboring trajectories approach and converge to the nominal trajectory. Hence, this kind of stability is usually referred to as ‘cyclic stability' or ‘orbital stability' [11].

The application of these principles to quadrupedal locomotion, however, has drawn far less attention. The only research conducted has been to analyze the implications of passive dynamics to true quadrupedal walking. Their principal finding was the identification of two stable passive dynamic walking gaits. The first was a two-beat gait in which the front foot and back foot move in phase, and second a four-beat gait in which the feet move 90° out of phase. The latter four beat gait is energetically more efficient, as lesser energy is lost in collisions. However this gait appeared to be stable only for a limited range of parameter variations.

1.4 What is Passive Dynamic Walking?

The founding principles of passive dynamic walking were first developed by Tad McGeer [12, 13] towards the end of the 1980s. At Simon Fraser University in British Columbia, McGeer demonstrated that a robot resembling a human and having a human-like form can walk down a slope without muscles or actuators. As opposed to traditional robots which expend energy by using motors to control every motion, Tad McGeer's earliest passive dynamic machines depended only upon gravity and the inherent natural dynamics of the swinging of limbs to move forward and hence down a slope.

The dynamic behavior of actuators, robots, or organisms when they are not drawing energy from a power supply (e.g., batteries, fuel, ATP) is termed as passive dynamics. Depending on the application, considering or altering the passive dynamics of a powered system can have extremely positive effects on performance, especially energy economy, stability, and task bandwidth. For mechanical devices that use no power source, their behavior is fully described by their passive dynamics. These devices are also commonly called as ‘passive’ devices.

1.5 Motivation for Passive Walking

The primary motivation for understanding and applying the concept of passive walking, is that it results in mechanical simplicity and high efficiency. Here, it is important to define a term called as 'specific cost of transport': *The specific cost of transport is equivalent to the amount of energy required to carry a unit weight over a unit distance.*

This specific cost of transport in humans is nearly about 0.2, whereas in other recent and rather conventional robots of today, for example, the Honda Asimo, the specific cost of transport goes as high as 3.23.

Therefore, the principal motivation for passive walking stems from the fact that the robots of today are highly energy intensive and not particularly efficient. Moreover, speed control and direction control is simplified when the details of generating a motion that mimics a trajectory are no longer a problem.

To consider an analogy, taking the case of the Wright brothers, whose premature attempts at understanding gliders, paved the way for manned flight and subsequent research into the field. Once they had a solid grasp over the subject of aerodynamics and control, adding a power source that could give thrust was only a small change. Similar to this notion, adding power to a passive walker involves only a comparably minor modification [14].

1.6 Gait Analysis

Four legged animals move around in a variety of ways. Depending on the size of the animal and the speed required by the animal, it chooses a gait from a variety of possible gaits. A gait is a repetitive pattern of leg movements used by an animal to move around. Gaits are normally distinguished in terms of footfall sequences, push sequence and the presence of suspension effect (normally in fast paced movement) [15].

The most common gaits are the statically stable walk, the amble, trot and gallop.

1. **Walk:**
   The walk is a four-beat gait characterized by triangular (3 leg) support which makes this gait statically stable. A four beat gait means that the 4 legs strike the ground at distinct

points in time. A sample pattern of movement in walk is right hind, right fore, left hind, left fore. In animals the fore legs lift just before the hind legs strike the ground so as to allow the hind legs to take the same foot position as that of the fore leg.

To remain statically stable, the body must shift its weight/COG so as to make sure that the COG lies within the support triangle formed by the three legs on the ground.

2. **Amble:**
   The amble is very similar to the walk and is essentially a sped up walk in which there will be usually only two supporting limbs. A supporting limb is a limb that is touching the ground.
3. **Trot:**
   The trot is a diagonal two beat gait. The animal moves a fore limb and the hind limb in unison. The pattern can be thought of as left diagonal (right hind and left fore), right diagonal (left hind and right fore).
4. **Gallop:**
   The gallop is an asymmetrical four beat gait in which the animal moves at fast pace. In this gait there is a lot of body rocking. There are also times at which there the animal is completely unsupported i.e. all the four legs are in the air.

Static gaits such as walk or crawl maintain static stability which means that the robot's center of mass is always within the polygon formed by its supporting legs. Dynamic gaits like the trot and gallop do not have this requirement. Although this allows for a faster locomotion, it comes at the cost of the need for a more difficult balancing act. In problems involving rough terrain where stability is the major factor, the static gaits are preferred over the dynamic ones.

1.7 Quadruped Motion Planning

To get collision free motions for the quadruped i.e. to get the quadruped from one state (initial position) to the next state (goal position), in static and dynamic environments is fundamentally a motion planning robotic task. The goal position and dimensions of obstacles is normally in a low dimensional *operational space*. In the case of quadrupeds, since the robot is meant to move in a 3D environment, the *operational space* is of dimension 3.

However when a feasible path is planned, a complete specification of the location of every point on the robot is to be known. The space which describes the complete specification of the robot at

a given point is called the *configuration space.* This *configuration space* represents the set of all transformations that can be applied to the robot given its kinematics. The greater the complexity and degree of freedom of the robot, the greater is the dimensionality of the *configuration space*, and hence greater is the difficulty of generating the space transformations. The fact that dimension of *configuration space* is equal to degrees of freedom of a robot is the biggest problem in the motion planning of a quadruped, since the quadruped often has a degree of freedom of around 12.

1.8 Quadruped Control Architecture

Most projects involving quadrupeds to traverse rough or unknown terrain, a multi-level planner and control technique is used. Such architectures have been verified on commercially available quadrupeds like the Little Dog and Sony's Aibo.

A typical 3 level planner/controller architecture looks the following:

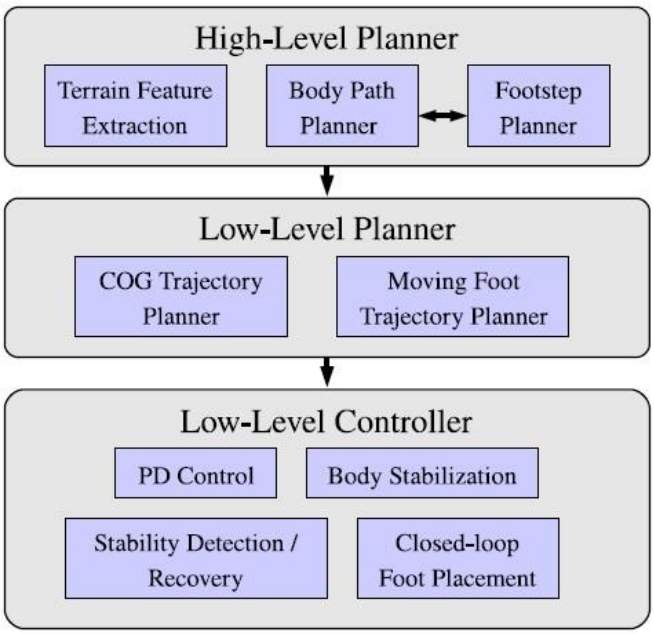


Figure 1.6 Typical 3-level control architecture in a legged robot [3]

In this architecture, the high planner performs the following tasks:

1. Generate collision and height maps from a 3D model of the terrain. The 3D model of the terrain can be obtained from the cameras or other sensors present on the robot or in a global frame.
2. Create a feature on maps of varying sizes around each point of the height map formed in the previous step. These feature maps would contain information like slope, height, average deviation of height etc.
3. Form a foot cost function i.e. a cost function which tells the feasibility and correctness of placing a foot at a given location. This would take into account the features at the point. The process of forming this cost function involves learning techniques.
4. From the foot cost map, the body cost map is generated. Then the minimum body cost path is planned between the start point and the goal point.
5. Now given the approximate best computed path, the footsteps the quadruped will approximately take is calculated. The calculation of these footsteps would involve information on the parameters of the gait used.

The low level planner then is used to determine the desired trajectory of the robot's COG. An important part of this is to ensure that the COG lies within the support triangle. The exact footsteps will be calculated and be given to the low level controller.

The low level controller takes care of the problems of fall detection and locally employs stability detection and recovery. The local controller stabilizes the robot by moving the COG into a statically stable zone during slips or sudden change in terrain. The low level controller also takes care of the close loop foot placement i.e. the robot doesn't deviate too much from the desired trajectory passed on to it by the low level planner.

1.9 Objective

The primary motivation of this work is to understand and then subsequently analyze a quadrupedal system which can incorporate the twin effects of motion planning and passivity. Work done in all laboratories till now has somehow only been restricted to one facet of the puzzle yet all the features have never been clubbed together to form a cohesive and unified solution.

The first objective would be to understand the depth to which research has been conducted into different subjects by virtue of an extensive literature review. That would provide the knowledge

and details of what has transpired till now and give a strong base on which one can deliberate on the topics that yet remain untouched and therefore, pursue them further.

The secondary objective would then be to capitalize on this extensive literature survey and effectively address the problems that remain unsolved through a logical and organized approach. With respect to stability and efficiency, there are various multitudes of finer details that researchers have failed to address. Therefore, as a starting step, kneed walkers have been analyzed and the ability to move in stable periodic gaits has been showcased. Even after being subjected to perturbations, the model reassumes its stable cycle thereby displaying its stability.

Thirdly, the final objective that would be taken up in the second phase of the project would be to design, analyze and thereafter develop a quadruped using a chronological approach. Following which, different gaits would be implemented on the quadruped and their stability and efficiency tested. Subsequently, other fairly complex concepts such as motion planning to traverse obstacles would be introduced that would aid in reaching the desired destination, along with a detailed analysis of object tracking that would allow such a quadruped to navigate more efficiently.

1.10 Organization of the Report

This report has been organized in a manner such that it is comprehendible. Following the introduction, the next section gives insightful details about the work done till now in different subjects of passivity and control and motion planning.

After a thorough literature review, the results generated using Matlab and Simulink would be utilized to produce stable cycles of a kneed walker, which then would be extrapolated to a quadruped in the future.

The next section expounds on designing the quadruped, which first involves assimilating and using simple dynamics of the quadruped to decide the requisite materials to be used. After which, sufficient analysis on gait simulation has been presented. Subsequently, CAD drawings have been made to make a blueprint of the robot that would be used while assembling it.

After constructing and properly assembling the robot, sufficient work has been presented regarding the experiments conducted on the quadruped with regard to different gaits and patterns, and also

incorporates paradigms of motion planning that have been applied on rovers to test the navigation capabilities.

Finally, the Conclusions and Future work are stipulated which would also dictate the work that could be pursued in the future.

Chapter 2

LITERATURE REVIEW

Robotic quadrupeds which can function as transportation machines present many advantages due to their high mobility and ability to traverse rugged terrain as opposed to conventional wheeled robots. However the high degrees of freedom and large number of actuators associated greatly increase their energy expenditure.

A large group of researchers are already actively engaged in working on Limit Cycle Walking, as will be shown in the following section. However, there is a small difference between passive walkers and limit cycle walkers with the former being the subset of the latter. Many robots are derivatives of the so-called Passive Dynamic Walkers [12], a subgroup of Limit Cycle Walkers.

In the first section, Passive Dynamic Walking robots will be addressed, followed by actuated Limit Cycle Walkers. Most robots using this concept are referred to as `Passive-based' walking robots, but it’s important to note that Limit Cycle Walking is a much more accurate description for them than the former.

2.1 Passive Dynamic Walkers around the world

Robots that depict a stable gait when walking down a gentle slope without actuation are called Passive Dynamic Walkers. The stance leg is essentially an unstable inverted pendulum and each step is considered to be a forward fall. The absence of any control or actuation distinctly defines Passive Dynamic Walking as a subset of Limit Cycle Walking.

The concept of passive dynamic walking was first introduced by McGeer, who initiated the idea to simulate and build 2D walkers with [12] and without knees [13]. Subsequently, Andy Ruina from Cornell University succeeded in demonstrating stable Limit Cycle Walking without the existence of any local stability by building a 3D uncontrolled toy that could walk but could not stand still [16]. Consequently, this group initiated research into a human-like 3D prototype with knees and arms [17]. This established the feasibility of the concept of passive walking for the development

of humanoid robots. In parallel, passive dynamic walkers have been developed throughout the world in various research groups [18, 19, 20].

1. **McGeer's Passive Dynamic Walker**:

   This was the first of the series of passive dynamic walkers. Pioneered by McGeer as has already been mentioned, this walker paved the way for future research enthusiasts to understand passivity.

   It consists of a simple mechanism wherein both the legs are hinged at the hip. This was the first machine demonstrate that a human-like form could actually walk down a slope if given some initial configurations with nothing but its own inertia.

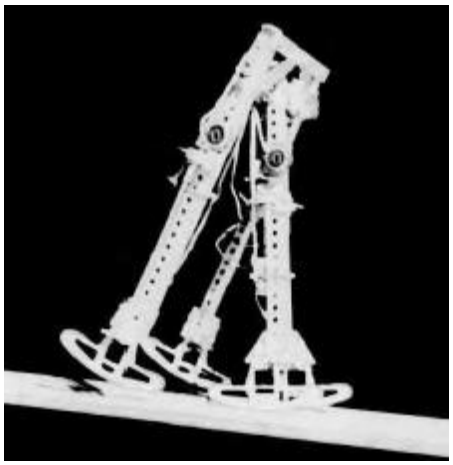


Figure 2.1 McGeer's straight legged passive dynamic walker [12]

2. **Cornell's McGeer copies:**

   What primarily started as an attempt to mimic the results of the previous research conducted by McGeer, led to further developments in the field of Passive Walkers, with Cornell's copy of these machines providing stable periodic motions along with knees. It consists of a swing leg and a stance leg wherein the hinges at the hip and the knee are free to move, therefore they are capable of generating a stable motion when given appropriate initial configurations. However, the link extending beyond the knee gets fixed to its upper part when it goes beyond its flexion. Therefore, there are two stages in the motion of this walker: one with the knee free thereby providing three links, the other with the knee fixed. So, the stable motion of this walker is divided into two parts – before knee-strike, and after knee-strike. Before knee-strike, it operates as a three link mechanism (one is the stance leg, and

two on the swing leg) and after knee-strike, it acts as a two link mechanism (one on the stance leg and the other on the swing leg).

Figure 2.2 Cornell University's copy of McGeer walking machines with knees [13]

3. **Cornell's tinker toy**:

   Developed by Andy Ruina and the lab at Cornell, these machines typically demonstrated how it is possible to have a stable limit cycle without any local stability at any given time at any given point.

Figure 2.3 Cornell's tinker toy that can walk but cannot stand still [16]

These tinker toys were so designed such that they would be unstable if kept still, however, when provided momentum, they could converge to a stable limit cycle without any control whatsoever. Hence, these mechanisms could balance only when walking but not when standing still, thereby giving the ultimate test of how it is possible to have the robot walk periodically without making it stable at any instant of time.

4. **Cornell's 3D walker**:

Again, developed at Cornell, this walker suggested that passivity as a concept could be pursued not through just symmetrical machines like previously developed. Possessing knees along with counter-swinging arms, this walker resembled a human more than any other machine developed till now.

Figure 2.4 Cornell's 3D walker with knees and counter-swinging arms [17]

The basic idea was to have the arms swing side-to-side at appropriate times to reduce side-to-side rocking. Also, the counter swinging of the arms was to reduce the angular momentum effects about the vertical axis.

2.2 Actuated Point/Arced Feet Walkers around the world

The largest group of typical Limit Cycle Walkers comprises of actuated robots with point feet. There is no surprise to this because they incorporate both the elements of efficiency and robustness. Even though they have actuation and control in their joints, however the resulting point or line contact and subsequently, the shape of their feet makes these systems under-actuated.

Also, since under-actuated systems in theory possess the capability to be stabilize locally (by virtue of the actuated joints, like the Acrobot [21]), this is not however seen in the robots mentioned ahead. They are primarily stabilized through cyclic stability and hence, categorically fall under the description of Limit Cycle Walkers. Here are some of the 2D/3D prototypes that used a four-legged symmetric construction (similar to McGeer's machines) or a guiding boom.

1. **Carnegie Mellon's 2D walker:**

   This robot's hip and knee joints are driven by direct drive electric actuators. However, the disadvantage of this is that trajectory control becomes practically impossible due to these

weak and highly back-drivable motors. Nevertheless, this robot was experimented on by ad-hoc controllers [22], through reinforcement learning [23] and with CPG control [24], hence, fully demonstrating the wide-ranging scope of Limit Cycle Walking.

Figure 2.5 CMU's 2D walker [28]

2. **'Rabbit' from CNRS and University of Michigan:**

   The robot, Rabbit's [25] hips and knees possess heavily geared electric motors. The control architecture is established on the concepts of hybrid zero dynamics and virtual constraints [26, 27]. All internal joints are subjected to tight control, which is performed using the angle made by the lower leg with the. Due to this, this machine has just one passive degree of freedom, and therefore the orbital or cyclic stability is computed only with one Floquet multiplier.

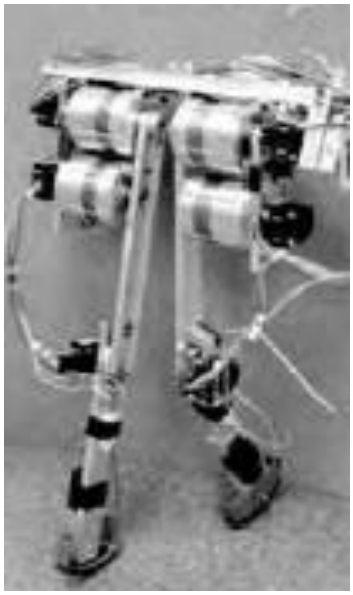


Figure 2.6 'Rabbit' [25] from CNRS, UMichigan

3. **'Mike' and 'Max' from Delft:**

   Between the two extremes of robots mentioned earlier, there are many two-dimensional Limit Cycle Walking robots that are pneumatically actuated – 'Mike' [29] and 'Max' [30].

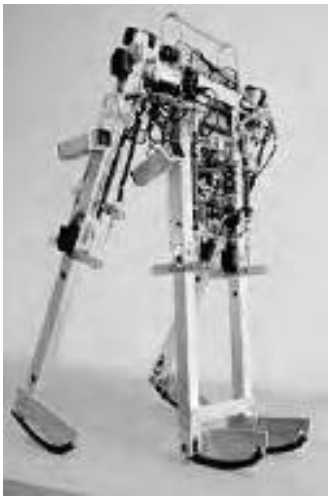


Figure 2.7 Mike [29] from Delft

4. **Cornell Ranger:**

   This semi-actuated walker currently holds the record of having the lowest specific cost of transport which is even lesser as compared to humans (0.20). Developed at Cornell, this biped has no knees and is capable of walking on level-ground. To steer the robot, radio control is used; the inner legs are twisted thereby allowing the robot to move laterally while walking forward. By improving the control and electronics, Ranger's energy use was again

reduced by 43% in 2010 and was able to walk 65 km in 2011 without any form of recharge or human touch.

5. **Cornell Biped:**

   Later, Cornell also built their 'Cornell Biped' which consisted of a torso, swinging arms, kneed legs and arced feet [2, 28]. However, number of degrees of freedom is only five due to internal mechanical couplings. One degree of freedom, of the hip joint and knee joints are completely passive, therefore limiting the only form of actuation to the ankle which performs a small push-off in the stance leg just after heel-strike at the swing leg. A spring is continuously loaded throughout one step by a small electric motor, which in turn executes the push-off force.

6. **'Toddler':**

   Belonging to MIT, it consists of two legs and two arced feet. One degree of freedom is at the hip joint which is fully passive, the other two degrees of freedom are at the ankle joint (roll and pitch) which are both activated by position controlled servo motors. Successful gait was obtained with fully feedforward ankle angle trajectories, hand tuned feedback and by applying reinforcement learning [31].

2.3 Actuated Flat Feet Walkers around the world

1. **Spring Flamingo:**

   Out of the limited number of flat feet Limit Cycle Walkers, 'Spring Flamingo' has been built at MIT and consists of seven links containing the torso, kneed legs and flat feet [32]. Series elastic actuators are used to actuate six internal joints. Here, geared electric motors are intentionally placed in continuance with springs [33]. This form of design potentially permits torque control on all the joints. Virtual Model Control, an intuitive approach to bipedal walking control, is used to compute the desired torques for all joints [32].

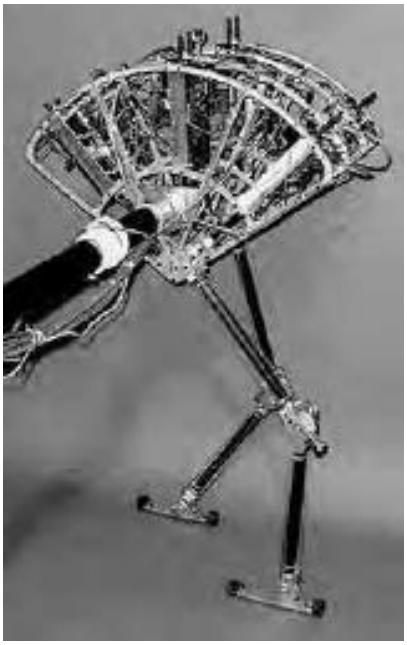


Figure 2.8 Spring Flamingo [32] from MIT

2. **Meta:**

   'Meta' is a prototype built for research at Delft University of Technology. It is similar in structure to 'Spring Flamingo', however the only difference being that the hip joint and the ankles have series elastic actuation whereas the knee joints are made passive. 'Spring Flamingo' is restricted to two dimensions by a boom construction and `Meta' by a symmetric construction of its legs. Currently there are no fully 3D flat feet Limit Cycle Walkers in existence.

2.4 State of the art on energy efficiency

The specific cost of transport or specific resistance is a term invariably used in determining the energy efficiency. This dimensionless number c_t defines the energy expended by a robot to travel some distance per weight of the walker [2]:

$$c_t = \frac{Used\ energy}{Weight\ *\ Distance\ travelled} \tag{2.1}$$

To distinguish between total system design efficiency, mechanical design efficiency and the applied controller forces, the specific energetic cost of transport c_{et} and the specific mechanical cost of transport c_{mt} are defined. The specific energetic cost of transport c_{et} considers the total amount of energy that the system has expended whereas the specific mechanical cost of transport c_{mt} deals only with the amount of mechanical work performed by the actuators.

To clearly categorize, the specific cost of transport is supposed to be measured at constant and comparable normalized walking speeds. Normalized walking speed is defined by the Froude number *Fr*, which is the actual walking speed divided by the square root of gravity (9.8m/s^2) times the leg length [2]:

$$Fr = {}^{v}/_{\sqrt{gl}} \tag{2.2}$$

Passive Limit Cycle Walkers are sufficiently efficient as compared to the other robots developed. For example, the first passive dynamic walker built by McGeer has a specific cost of transport of approximately 0.025 while walking at a Froude number of Fr = 0.2 [2]. In comparison, the specific energetic cost of transport c_{et} of humans is about 0.2 whereas the specific mechanical cost of transport c_{mt} of humans is estimated at about 0.05 [2]. Honda's humanoid Asimo, which is supposed to be state of the art is estimated to have c_{et} = 3.2 and c_{mt} = 1.6 [2].

The Actuated Limit Cycle Walkers demonstrate mechanical costs of transport that are almost on par with the passive dynamic walkers: c_{mt} = 0.06 for 'Dribble', c_{mt} = 0.055 for the 'Cornell Biped', c_{mt} = 0.08 for 'Denise' and c_{mt} = 0.07 for 'Spring Flamingo' all at a speed of about Fr = 0.15 [2]. On the other hand, the specific energetic cost of transport c_{et} for these machines varies greatly, typically depending upon the effort put into limiting the overhead energy expenditure of the system. It ranges from c_{et} = 0.2 for the 'Cornell Biped' to c_{et} = 5.3 for 'Denise' [2].

2.4.1 Conclusion on Literature Review of Limit Cycle Walking

As a conclusion from the extensive articles reviewed on passive dynamics and limit cycle walking, it can be noted that the energy performance of robots significantly increases when the concept of Limit Cycle Walking is introduced as can be seen in various bipedal robots that have been built over time.

2.5 Popular existing Quadrupeds

1. **StarlETH from ETH Zurich:**

   This project, initiated by Christian David Remy from ETH Zurich aims at designing and constructing an energy efficient robotic quadruped which is capable of vast multitudes of dynamic motions such as walking, trotting, etc. While the work is currently in progress, the objective is to come up with fundamentally new control strategies that incorporate the advantageous characteristics of passive dynamics into an optimal controller. This quadruped has highly compliant elastic legs.

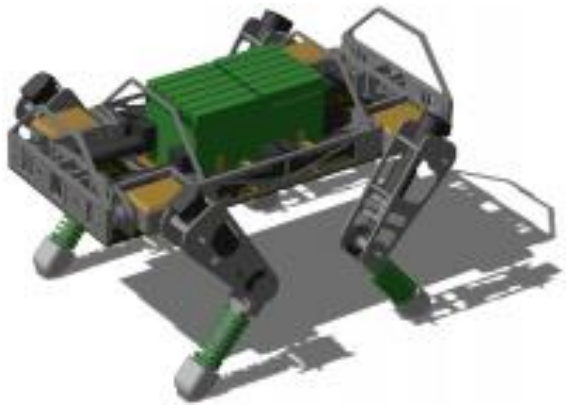


Figure 2.9 StarlETH [34], a quadruped robot by ETH

2. **BigDog from Boston Dynamics**

   BigDog [35] is a rough-terrain robot that walks, runs, climbs and carries heavy loads. The power is derived from an engine that drives a hydraulic actuation system. It has four legs that are articulated like an animal's and also has consists of compliant elements to absorb shock and subsequently, recycle energy from one step to the next. However, the most appealing factor is the size: about 3 feet long, 2.5 feet tall and weighs 240 lbs.

   The main disadvantage like any other powered/actuated quadruped robot is its inherent incapability to utilize the natural dynamics and incorporate an efficient gait. Even though it has the ability to walk 20miles without refueling, yet that feat would most certainly go to its bulky engine and excessive performance specifications which are not all efficient.

2.6 State of the art on passivity, control in quadrupeds

Research literature on passivity and control in quadrupeds is fairly minimal; however, there has been a sudden upsurge of activity and investment in trying to understand the dynamic motion of these machines.

Martin Buehler conducted research [36] on the Scout 2 quadruped which was primarily to ascertain the passive dynamics on quadrupedal bounding and also stable running using two control strategies. According to these experiments, it was reconfirmed that stability improves at higher speeds. The group earlier conducted experiments on understanding the galloping-gait of quadrupeds with compliant legs followed by attempts at understanding its natural gaits [37].

Roland Siegwart and the ASL group in ETH Zurich initiated research into the stability analysis in a quadruped [38]. This numerical simulation examined the influence of various parameters on the stability and locomotion speed. This was followed by attempts at understanding the walk and trot-gait of quadrupeds that would allow a unified model to be applied for both these gaits together [39].

Figure 2.10 Scout 2 [36], a quadruped robot by McGill University

Furthermore, his group conducted research into spring-leg systems for quadrupeds that exhibit passivity as well [40]. The efficiency of running gaits in nature results from a passive elastic oscillation on springy legs. They applied this principle to robotic systems by endowing them with high compliance series elastic actuators in which the electric motors are decoupled from the joints by elastic elements. Further improvisations were done using springs in ankles to reduce energy lost due to damping and during impact with the ground [41]. It also simultaneously allowed for a natural-looking stance configuration wherein actuator input and ankle spring properties were

optimized. This work was then carried forward by incorporating torsional springs at the hip wherein two compliant robotic legs were compared, that are able to perform precise joint torque and position control, enable passive adaption to the environment, and allow for the exploitation of natural dynamic motions [42].

Recently, their group has experimented with the newly designed ALoF [43] and StarlETH [34] robots, wherein a practical implementation of the principles propounded was sought in order to minimize energy consumption and increase robustness. StarlETH is an upgraded version of ALoF that uses high compliant series elastic actuators.

Also, legged-robot walks on irregular terrain such a damaged area or collapsing footholds have been simulated [44]. In this strategy, a locomotion method has been proposed in which the quadruped robot does not stumble on weak and irregular slopes.

2.7 Quadrupedal Motion Planning

This survey focuses on the methods commonly used for motion planning and control. First methods dealing with static situations are described. Then the expansion of these methods to handle dynamic environments is described.

2.7.1 Roadmaps (Combinatorial)

To understand the problem of motion planning, the 2-dimensional configuration space is considered. For such low dimensional space, a strict deterministic approach that exploits geometrical properties of the configuration space can be applied. These approaches built something known as the combinatorial roadmaps [45] that capture all information required for planning. One such approach is called the shortest path roadmap which is illustrated in fig 2.11.

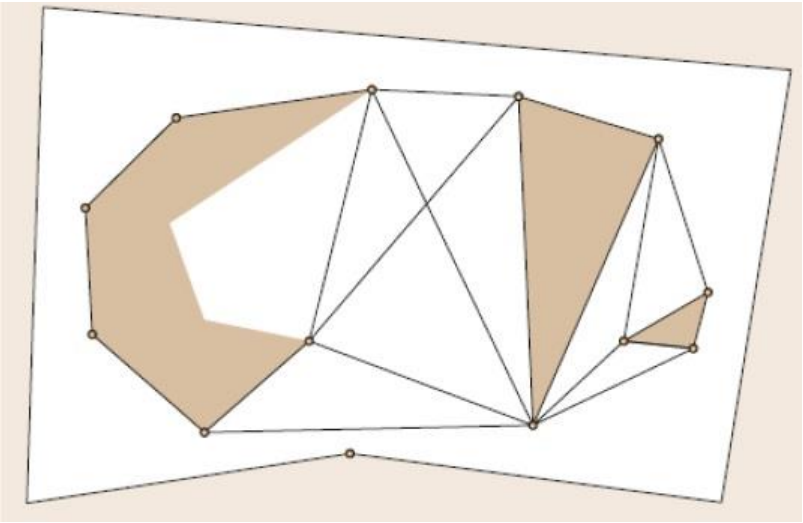


Figure 2.11 Shortest-path roadmap example. [45]

A graph is formed with nodes on the obstacle vertices and an edge exists when a pair of vertices is mutually visible. The fact that feasible paths may touch obstacles must be considered during the obstacle modelling.

The technique of cell decomposition is demonstrated in fig. 2.12. Here, the collision free configuration space is decomposed into cells shaped like trapezoids or triangles. At the middle of every cell, a roadmap node/vertex is placed and edges are formed between adjoining cell nodes. The collision free path can then be calculated with a graph search algorithm.

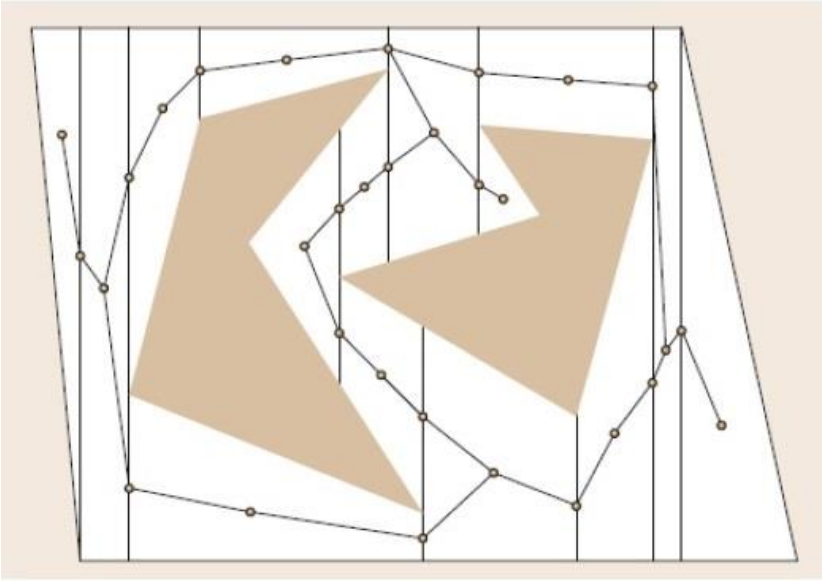


Figure 2.12 Example of space vertical decomposition.

Another method of cell decomposition which derives a lot of its principles from topology is the Voronoi Cell decomposition. Mathematically a Voronoi diagram is a way of dividing space into a number of regions. For a 2D case the segments of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest sites. Hence by placing a graph node at the middle of each Voronoi cell one can create a connectivity graph and solve the motion planning problem [46, 47].

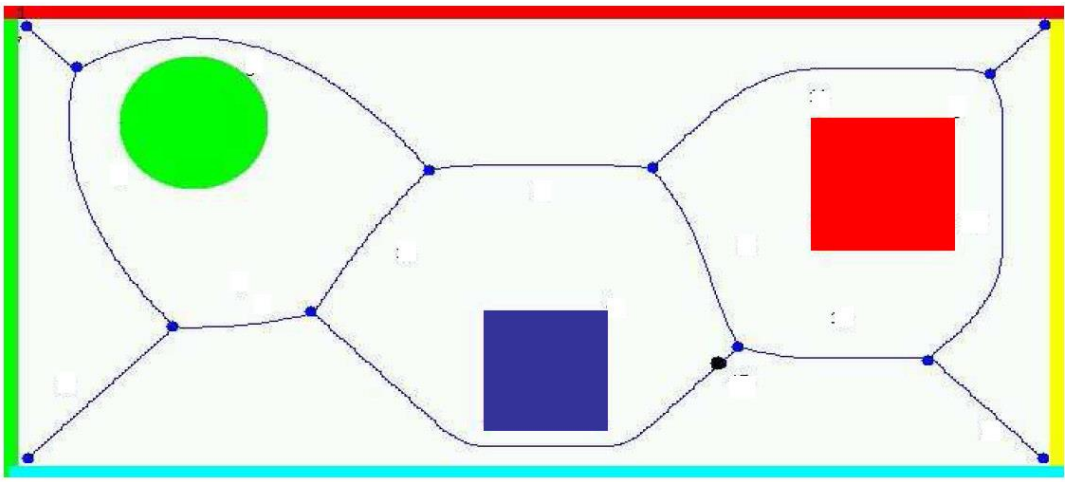


Figure 2.13 Example of a Voronoi diagram with the circle and square type obstacles [47]

The completeness that these deterministic algorithms provide is its main advantage. However for a high dimensional space this need for completeness is the main weakness. Due to combinatorial state explosion, these algorithms are ineffective for high dimensional configuration space. The main use of these algorithms is actually in local planners which are a part of a more complex planner.

2.7.2 Sample based planning (probabilistic)

To overcome state explosion problems, sample based methods were designed. For complex operations these methods are the most common choice. The completeness is not deterministic in these methods i.e. they will find a solution if it exists, however it may run forever if the solution does not exist. To satisfy probabilistic completeness, the probability that the method finds a solution tends to one if a solution exists. This concept of probabilistic completeness is often used a parameter to differentiate between different types of planners.

The sample based planners are of two types: *multi-query planners* and *single query planners*. For multi query planners, a roadmap representing connectivity of configuration space is generated and later multiple path search request can be processed. The procedure to develop the roadmap is

1. Choose sample configurations and develop a model of the configuration space

2. For configurations that does not collide with obstacles a node is created in a graph with collision free configurations. Paths between these nodes are the edges.

3. By connecting the new nodes with the pre-existing ones around a given neighborhood, the local planner is able to connect the entire collision free configuration space.

This algorithm follows the approach called probabilistic roadmap (PRM). This can be seen in fig 2.14.

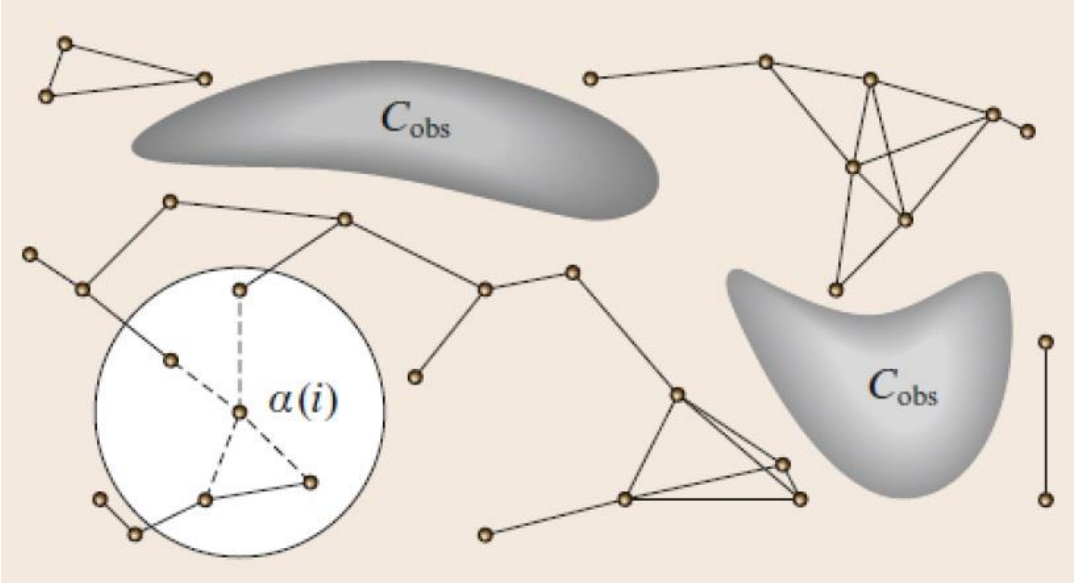


Figure 2.14 Example of sampling-based roadmap construction. The process goes incrementally by attempting to connect each new sample (α(i)) to the vertices in the roadmap within a certain distance from the new sample. [45]

The main difference between the various sample based planners is the method with which new sample configurations are chosen. The very first version which was described above used random selection of samples [48]. However since there is no guarantee that these random points will give more information about the configuration space, this method is not effective. In [49] it was proposed to put samples close to obstacles. This leads to a roadmap similar to the shortest path roadmaps. For sensitive operations [50] it was suggested to place samples far away from obstacles

to reduce chances of collision. This leads to a roadmap similar to that or a Voronoi graph. In [51] a deterministic sampling was used which places nodes on a predefined grid. The comparison of various multi query methods was presented in [52]. It was shown that the best performance is achieved when the probabilistic and deterministic approaches are combined. An interesting method proposed to increase efficiency was to decompose the general problem into low dimensional workspace problem. Then to sample, obstacle points are chosen and then these points are mapped to the initial high dimensional configuration space. This method lowered the computational cost of the planning.

Sample based methods of single query type do not build roadmaps to represent the whole space. Whenever it searches for a solution, it generates a tree like graph which targets time efficiency. These trees grow to satisfy the objective of connecting the initial and final configurations of a given problem. Rapidly Exploring Dense Tree (RDT) or Rapidly Exploring Random Tree (RRT) was first presented in [53]. These trees were shown to give good performance and are widely used in much robotics applications. Fig 2.15 shows the pseudo code of this tree generation. After the initialization in step 1 the RDT algorithm works incrementally. In step 3 a random collision-free configuration is being chosen and in step 4 an already existing graph vertex (qnear), which is closest to this new sample, is found. In step 5 the NEW_CONF function selects a new configuration (qnew) by moving from qnear by an incremental distance of Δq in the direction towards qrand. In steps 6 and 7 a new node and a new edge are added to the existing graph. To ensure that the tree will finally converge to the goal, the goal configuration is chosen as a probabilistic qrand with a certain rate. Fig. 2.16 provides illustration on how the algorithm explores the space.

BUILD_RDT(q_{init}, G, Δq)

1. G.init(q_{init});
2. **for** $k = 1$ **to** K
3. $q_{rand} \leftarrow$ RAND_CONF();
4. $q_{near} \leftarrow$ NEAREST_VERTEX(q_{rand},G);
5. $q_{new} \leftarrow$NEW_CONF(q_{near}, Δq);
6. G.add_vertex(q_{new});
7. G.add_edge(q_{near},q_{new});
8. Return G

Figure 2.15 Rapidly-exploring dense tree construction algorithm. Here G is the graph storing the configurations. Other notations are explained in the cited paper [53]

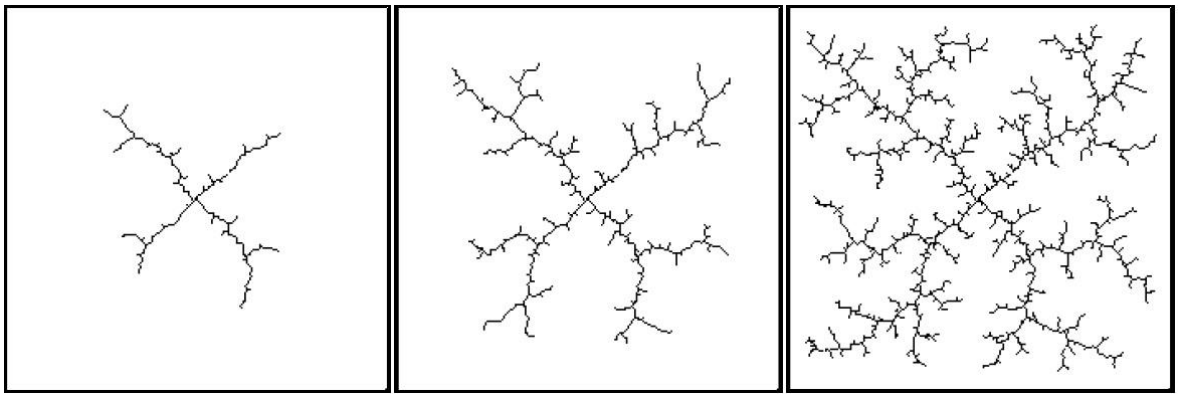


Figure 2.16 RDT construction example. [53]

The variations of these algorithms are distinguished based on the number of trees that are grown from each node. Unidirectional trees have only one tree while bidirectional trees [53] have two trees one growing from start configuration and the other one that grows form the goal configuration. In [54] multidirectional methods use more than two trees.

An interesting modification of the RRT is presented in [55]. By building tree like graphs without choosing new nodes randomly, predefined motion primitives are used to obtain unique sample

configurations. These configurations are then assigned a cost value which depends on the closeness to obstacles and the length of the path required to reach the configuration from the initial one.

In many cases with the quadrupeds, dynamic constraints need to be met. These often non holonomic constraint planning is found in [56] and [57]. These methods are designed for systems with complex dynamics that are described by physical models instead of equations of motion. Construction of the 'Search trees' is exploited in these methods. A new sample configuration qrand is not selected explicitly but instead a control law (for e.g. a vector of motor torques) is chosen randomly from a bounded region and applied to one of the nodes that is already present in the tree. Then the robot's forward dynamics is simulated in accordance to this control law until a collision occurs, or a state constraint is violated, or if a maximum number of steps is exceeded. In the last case a new vertex in the tree is formed.

However, since usually integrating the equations of dynamics of complex systems are computationally demanding, the search should be guided to reduce the amount of computerized calculations. Authors of [56] use deterministic approach to choose which vertex should be expanded. This method is called Planning by Interior-Exterior Cell Exploration (KPIECE). The KPIECE approach can be seen in fig. 2.17

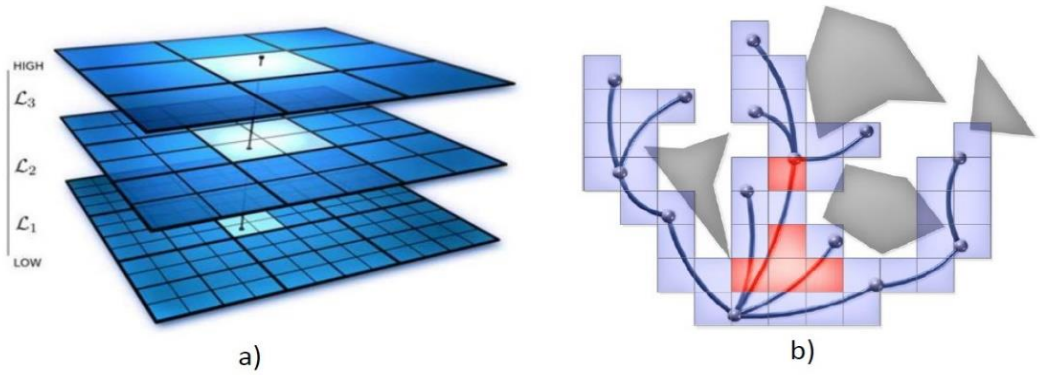


Figure 2.17 Grid structure of KPIECE approach. a) Multilayer grids for space discretization. [19] b) Representation of a tree of motions and its corresponding discretization. Cells are distinguished into interior (red) and exterior (blue). [56]

In [57] forward simulation of system dynamics for planning is used. They use symbolic action planners (usually used in artificial intelligence) instead of forming complex grid structures. The symbolic action planner guides the search and identifies regions of space the planner should explore further.

2.7.3 Feedback planning

Feedback planners are an approach which explicitly accounts for the imprecision in information presented. This imprecision manifests itself in practical dynamic environments. The environment may change during motion execution and the results of the motion bring about an uncertainty in robot state. Such planners usually construct a potential function in the allowable state space without specifying a restrictive path. The robot then moves in a direction where potential function decreases.

The classical potential field method (PFM) can be visualized by assuming that the robot is a particle that moves in the configuration space under the influence of a force field. Like in electrostatics and magnetics, a field has attractive components which attract a particle towards a goal and some repulsive components which the particle away from obstacles (fig 2.18).

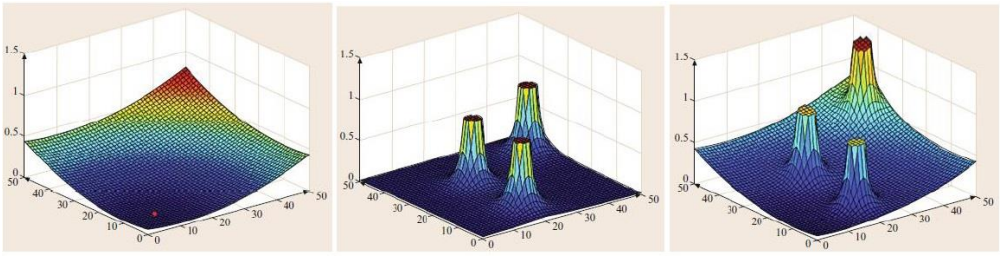


Figure 2.18 Attractive, repulsive and resulting potential functions.

In [58] by calculating resulting forces for multiple points on a robot the PFM was applied to robots. For a fast moving point the region where the repulsive forces act is enlarged to provide space for deceleration. Hence the current speed of the operating point often alters the potential field.

The biggest disadvantage of this method is the presence of local minima especially in the presence of concave obstacles. Another demerit is the need to select appropriate potential functions. If the potential function chosen is not carefully done, it would be impossible for a robot to move between closely spaced obstacles. The presence of multiple obstacle was seen to create oscillations in robot motion as well.

The vector field histogram method (VFH) proposed in [59] avoids the negative effects of the PFM. The two stage process first calculates obstacle density in all directions where the robot is the centre. The direction with lowest obstacle density becomes candidates for continuing movement. In the second stage, one of the candidate directions is chosen to move and reach the goal. Another technique is the use of the navigation function. This navigation function is basically a potential

function free of local minima [60]. By using several local potential functions a global construction algorithm constructs the navigation function.

2.7.4 Dynamic Environment Planning

The methods described before are designed to work in static environments and are not suitable to handle moving obstacles. After the configuration space is constructed, if any obstacle moves the entire computation of configuration space needs to be repeated. For quadruped designed to move in dynamic environments, decisions are need to be made in bounded time. Hence the following literature will help develop fast algorithms to motion plan for a quadruped in a dynamic environment.

In [61] spatial and temporal planning problems are separated. This method was developed for movements inside factory floors where multiple robots are constrained to move along pre-specified path networks. Since geometrically the robot moves only along edges of pre-computed roadmap (like the lines painted on the floor), moving obstacles (like other robots moving around) are avoided only in time by tuning the robot's velocity.

For an unstructured and unspecified environment no universal solution has been obtained. Most approaches adapt RRT search for dynamic environments. In [62] Anytime RRT* was presented which finds a feasible but not optimal solution and allows the robot to start executing the path. The beautiful part of this algorithm is that as the robot is moving along the first piece of path, the remainder part is still being optimized according to data loaded at the beginning of planning or data which is just loaded during the movement. This method often results in a time optimal solution.

The concept of global planning with local re-planning simplifies motion planning in dynamic environments. When an obstacle moves and obstructs the path of the robot, only a small stretch of path requires re-planning. By only performing re-planning locally the robot can get back on track quickly. In [63] such a method based on the PRM is proposed. At the beginning a classical PRM roadmap for static objects is created and stored. When the roadmap is obstructed by a moving object, some of the edges and vertices of the roadmap get invalidated. During this point a query is sent to the roadmap. Several scenarios are possible at this point depending on the extent of obstruction. Fig. 2.19 provides visualization on this planning.

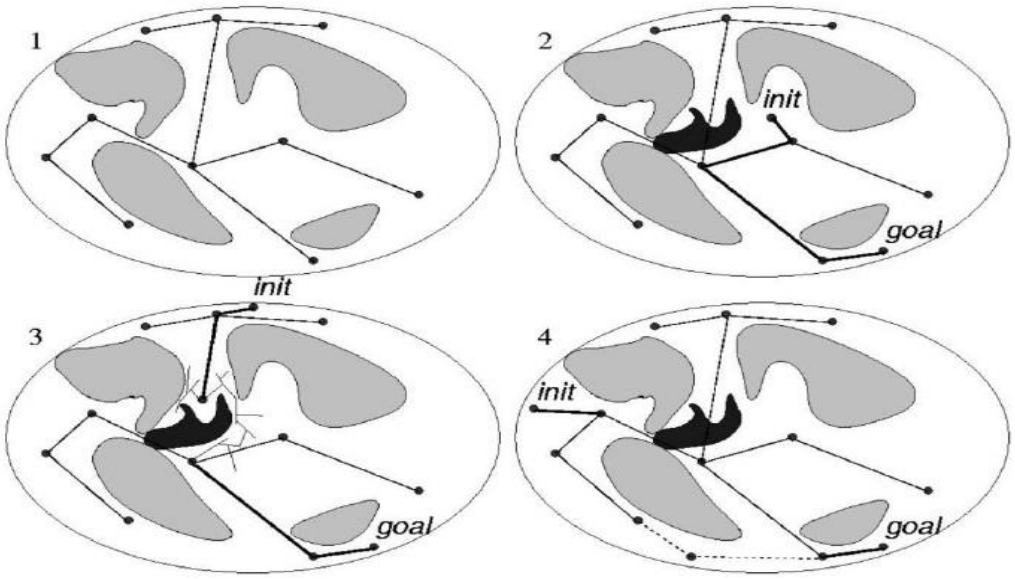


Figure 2.19 Work of PRM-based planner for dynamic environment [63]

A way to handle dynamic environment by not re-computing the paths is by just deforming them when an obstacle approaches. This was first implemented in the concept of *elastic bands* [64]. The framework corresponds to its name as from a visual point of view the path appears as an elastic band deformed by a moving obstacle. For this each piece of the path is created by two forces: an external force pushes it away from obstacles while internal force tries to make the path shorter. A similar approach applied to a nonholonomic mobile manipulator is described in [65]. The authors of [66] found that *elastic bands* can be complemented with the time dimension. Hence now the trajectory can be deformed not only geometrically, but also temporally. This clever adjustment allows the robot to stop while to letting an obstacle to pass by and then continue motion along the same path. Reactive Deforming Roadmaps (RDR) presented in [67] is an expansion on the elastic bands. In this the whole roadmap reacts to changes in the environment and adapts to them in real time. This is illustrated in fig 2.20

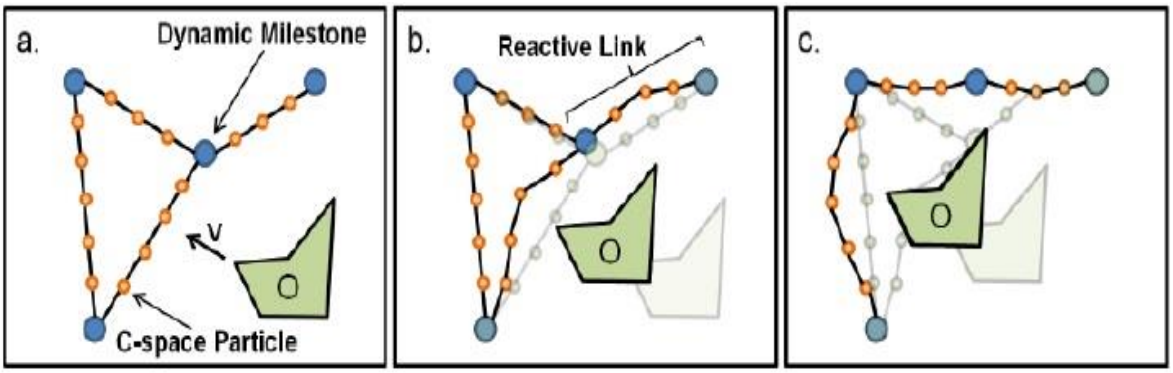


Figure 2.20 Reactive Deforming Roadmaps (RDR). (a) The RDR contains a set of dynamic milestones and reactive links. (b) As the obstacle O moves, the dynamic milestones move as well and the reactive links of the roadmap deform to avoid the obstacle boundary.

Leven and Hutchinson in [68] and Kunz et al. in [69] present dynamic roadmaps (DRM). This is basically a PRM specially arranged and altered to support unstructured dynamic environments. The idea is that planning is divided over many episodes. Each episode then contributes to a cost of planning. This method requires intensive preprocessing which is computationally hard. Hence the method is divided into offline pre-computation and online path search.

2.7.5 Conclusion on Literature Review of Quadrupedal Motion Planning

Out of the two biggest paradigms of current dynamic environment planning, elastic roadmaps is a universal, intuitive and powerful approach while DRM is probabilistically complete. Since the very nature of the framework for elastic roadmaps makes it incomplete and it may provide non optimal solutions in cluttered environments with complex-shaped obstacles, the DRM is better to use for complex shaped obstacles. However for a quadruped which requires quick computations since the cost of instability is too high, time of computation plays a big role. The necessity of offline computation in DRM is hence unattractive for a quadruped planning.

Chapter 3

WORK ON A KNEED BIPEDAL WALKER AND COMPARISON WITH QUADRUPEDS

3.1 Why Kneed Walkers?

Robotic quadrupeds which can function as transportation machines present many advantages due to their high mobility and ability to traverse rugged terrain as opposed to conventional wheeled robots. However the high degrees of freedom and large number of actuators associated greatly increase their energy expenditure. This problem can be overcome by exploiting the natural or passive dynamics of the system in order to maximize the energy efficiency. The analysis of passive dynamics involves the derivation of the dynamic equations of motion which are then numerically integrated to generate the trajectories of the states of the system. The main proposal is to create a dynamic model of a quadruped and generate optimal starting configurations by simulating the quadruped on Simulink/MATLAB and using the SimMechanics Toolbox. The method is verified by analysis of a two legged kneed walker.

The greatest challenge with analyzing the passive dynamics is absence of extensive literature on using natural dynamics on quadrupeds. The necessity of analyzing the passivity of bipedals is hence justified to confirm the concepts that are being implemented on a quadruped. By comparing the existing methods with the methods that are being proposed, the attempt is to show the similarity in results of the methods. In entirety simulating a bipedal on Simulink to understand its passive dynamics may seem to be a futile effort given the presence of an analytical solution; however this allows one to confirm the extension of this simulation based procedure for a quadruped.

3.2 Kneed Walker Model

The kneed walker model is essentially an under-actuated system placed on a ramp having a downward incline angle γ as seen from the horizontal plane.

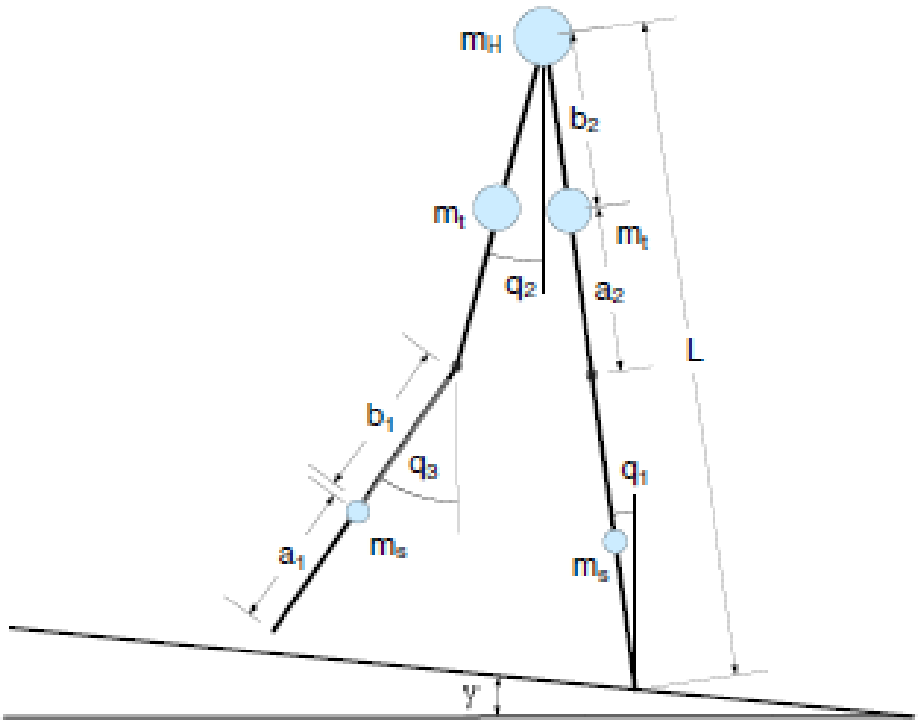


Figure 3.1 The Bipedal Kneed Walker [70]

From Fig 3.1, a clear picture of all the angles that are defined from the vertical axis is visible. All these angles are defined globally. Both legs have two point masses, m_t for the upper leg or thigh and m_s for the lower leg or shank. There is also a central mass inserted at the hip which is m_H. The link lengths as can be seen from the figure are defined as follows:

$$\text{L} = l_t + l_s, \quad l_s = a_1 + b_1 \text{ and } l_t = a_2 + b_2$$

At the commencement of each step, the stance leg is considered as a single link of length L, while the swing leg is assumed to be composed of two links which are connected by a frictionless joint at the knee. Unlocked knee dynamics dictate the dynamics of the system until the swing leg which is composed of two links comes forward and straightens. As soon as the swing leg is fully extended,

a discrete event termed as *knee-strike* occurs. Due to the collision, the velocities change instantly and as a consequence, the model becomes a two-link system dictated by its locked-knee dynamics.

This locked-knee dynamic system persists in this phase until the swing foot hits the ground. This collision is termed as a *heel-strike* event which is again, like *knee*-strike accompanied with velocity changes. After *heel-strike*, the model reverts to the initial unlocked-knee dynamics with the stance leg becoming the swing leg and vice-versa. Figure 3.2 clearly illustrates the entire step-cycle.

It is to be noted that if *heel-strike* occurs before *knee-strike* that is, if the swing leg hits the ground with a bent knee, then the robot will not be able to recover passively and stabilize itself. Hence, in that case, this particular event would be classified as a failure.

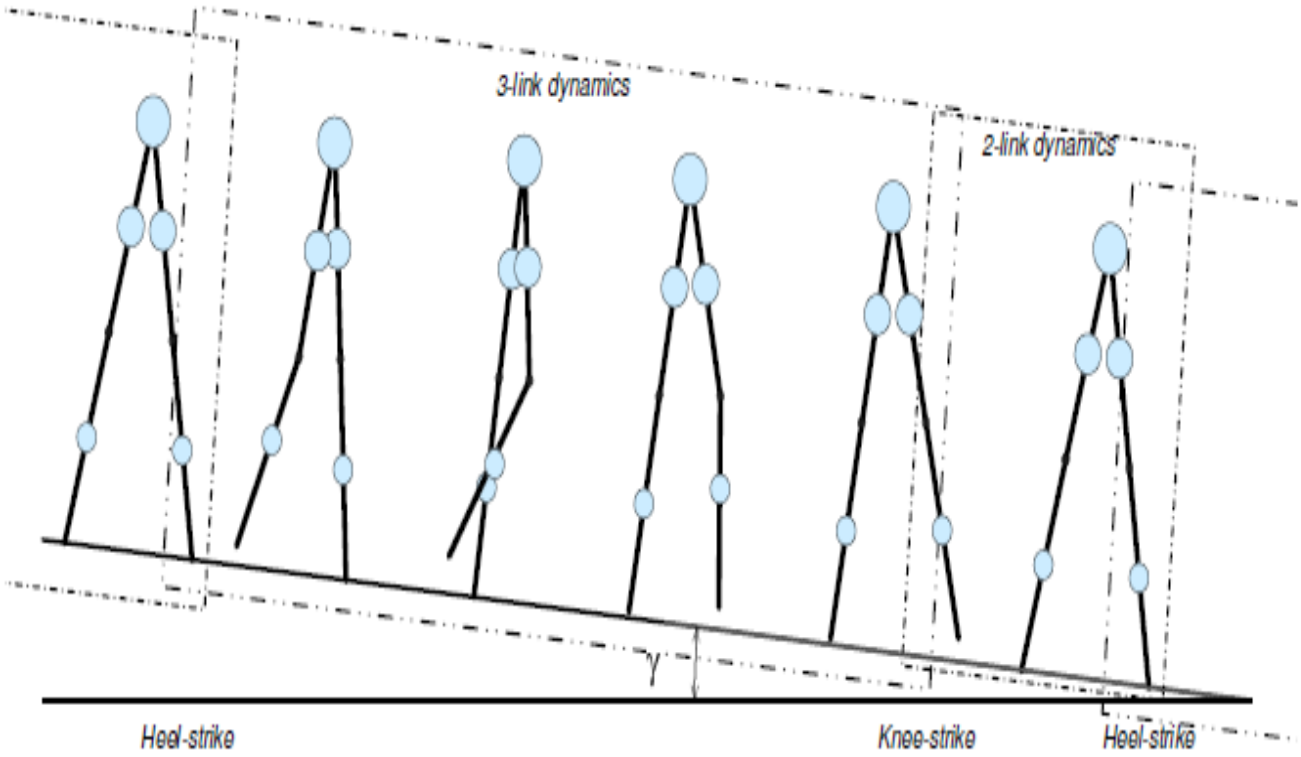


Figure 3.2 Step cycle for the Kneed Walker [70]

3.3 Dynamics of the Kneed Walker

3.3.1 Unlocked Knee Dynamics

For the unlocked-knee dynamics, that is when the swing leg is unlocked, the system is essentially a three-link pendulum. Lagrangian formulation can be used to derive the full equations of motion. Planar manipulator dynamics are shown in the standard form in the first equation followed by the specific inertia, velocity-dependent and gravitational matrices in the following equations.

$$H(q)\,\ddot{q} + B(q,\dot{q})\,\dot{q} + G(q) = 0 \tag{3.1}$$

$$\mathrm{H} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{22} & H_{23} \\ H_{13} & H_{23} & H_{33} \end{bmatrix} \tag{3.2}$$

$$\mathrm{B} = \begin{bmatrix} 0 & h_{122}\dot{q}_2 & h_{133}\dot{q}_3 \\ h_{211}\dot{q}_1 & 0 & h_{223}\dot{q}_3 \\ h_{311}\dot{q}_1 & h_{322}\dot{q}_2 & 0 \end{bmatrix} \tag{3.3}$$

$$\mathrm{G} = \begin{bmatrix} -(m_s a_1 + m_t (l_s + a_2) + (m_h + m_s + m_t)L)\,g\,sin\,(q_1) \\ (m_t b_2 + m_s l_t)\,g \sin\,(q_2) \\ m_s b_1\,g\,sin\,(q_3) \end{bmatrix} \tag{3.4}$$

Where,

$$H_{11} = m_s {a_1}^2 + m_t (l_s + a_2)^2 + (m_h + m_s + m_t)L^2 \tag{3.5}$$

$$H_{12} = -(m_t\ b_2 + m_s\ l_t)L \cos(q_2 - q_1) \tag{3.6}$$

$$H_{13} = -(m_s\ b_1)L \cos(q_3 - q_1) \tag{3.7}$$

$$H_{22} = m_t {b_2}^2 + m_s {l_t}^2 \tag{3.8}$$

$$H_{23} = (m_s\ l_t\ b_1)\cos(q_3 - q_2) \tag{3.9}$$

$$H_{33} = m_s\ {b_1}^2 \tag{3.10}$$

$$h_{122} = -(m_t\ b_2 + m_s\ l_t)L\sin(q_1 - q_2) \tag{3.11}$$

$$h_{133} = -(m_s\ b_1)L\sin(q_1 - q_3) \tag{3.12}$$

$$h_{211} = -h_{122} \tag{3.13}$$

$$h_{223} = (m_s\ l_t\ b_1)\sin(q_3 - q_2) \tag{3.14}$$

$$h_{311} = -h_{113} \tag{3.15}$$

$$h_{322} = -h_{233} \tag{3.16}$$

3.3.2 Locked Knee Dynamics

As soon as knee-strike occurs, the knee becomes locked and so the swing leg becomes one unified link without any frictionless joint at the knee. Therefore, following the knee-strike, the dynamics are represented by straight legs. The dynamics resemble those of the compass gait biped but with slightly different configuration of masses. Again, Lagrangian formulation is used to derive the following equations and they have been shown ahead for completeness.

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \tag{3.17}$$

$$B = \begin{bmatrix} 0 & -h\,\dot{q_2} \\ -h\,\dot{q_1} & 0 \end{bmatrix} \tag{3.18}$$

$$G = \begin{bmatrix} -(m_s\,a_1 + m_t\,(l_s + a_2) + (m_h + m_s + m_t)L)\,g\sin(q_1) \\ (m_t\,b_2 + m_s\,(l_t + b_1)g\sin(q_2) \end{bmatrix} \tag{3.19}$$

Where,

$$H_{11} = m_s a_1^2 + m_t (l_s + a_2)^2 + (m_h + m_s + m_t)L^2 \quad (3.20)$$

$$H_{12} = -(m_t b_2 + m_s (l_t + b_1))L \cos(q_2 - q_1) \quad (3.21)$$

$$H_{22} = m_t b_2^2 + m_s (l_t + b_1)^2 \quad (3.22)$$

$$h = -(m_t b_2 + m_s (l_t + b_1))L \sin(q_1 - q_2) \quad (3.23)$$

After the swing foot collides with the ground, another discrete event is said to take place called the heel-strike after which, the stance and swing legs are switched. This whole process completes a full step of the kneed walker.

3.4 Simulink based model for a bipedal's passive dynamics

The previously proposed methods to analyze the stability of natural gaits involve several assumptions like:

1. Completely inelastic collisions when the foot hits the ground.
2. No friction or damping at the joints
3. Negligible scuffing of the swing leg foot during swing

Apart from using such assumptions, these methods require a prior knowledge of the dynamic equations of motion to carry out numerical integration and subsequent stability analysis. When attempting to extend such techniques to a high degree of freedom machine like the Quadruped, the derivation of the equations of motion and integration of machine states become quite complex and involved. To overcome this problem, a simulation based approach where the robot is simulated in a virtual environment and physics solvers estimate the time varying states of the robot, has been proposed. The simulations in this thesis have been carried out using the SimMechanics toolbox in Simulink/MatLab.

Some of the issues involved in carrying out a simulating such a robot on Simulink are discussed below.

- **Simulation of Ground contact:**

  Ground contact simulation can be done using either the Hard stop contact or Soft stop contact. These contacts can be simulated by assuming a combination of springs and dampers between the robot foot and the ground which are activated once the foot touches the ground. The springs represent the elastic nature of the collision while the dampers simulate energy dissipation and inelastic effects. An example of such a contact can be seen in Fig. 3.3.

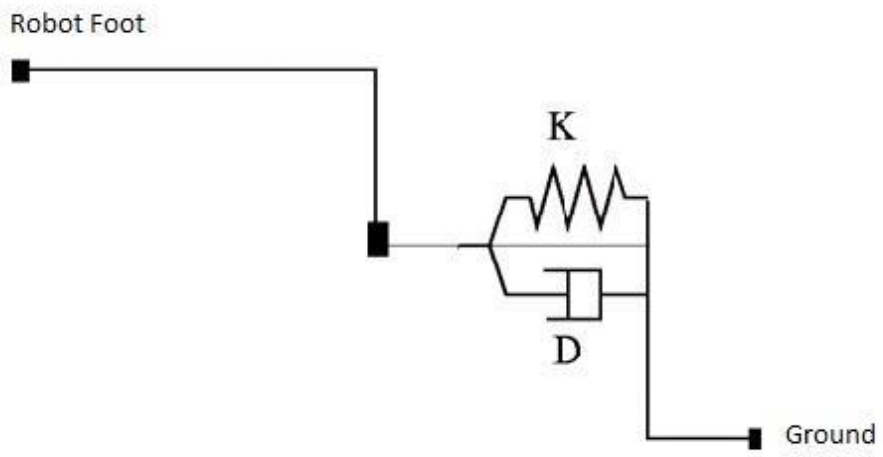


Figure 3.3 Hard contact

- **Simulation of Knee Extension (Knee Strike):**

  To simulate Knee Strikes, a Hard stop contact between the Thigh (upper leg) and Shank (lower leg) is simulated, which gets activated during hyper extension of the knee joint. This is similar to the simulation of ground contact.

- **Simulation of Inclined slopes:**

  By varying the gravity vector as a function of time or state of the robot, changing slopes of the terrain can be accounted for.

- **Simulation of Friction:**

  Friction is simulated by using Body Actuator blocks in Simulink. For the joints, joint stiction actuators are used.

After creating the model for the Robot in Simulink, simulations are carried out with various initial configurations. A good initial configuration is one in which the robot is able to complete several walk cycles without falling. It can be known if the robot has fallen or not by checking if the Center of Gravity of the robot has fallen below a predetermined threshold. By optimizing the initial configuration of the robot an optimal starting configuration can be found which gives a reasonably stable natural walk.

A record of the stable configurations and stable state cycles for passive walking in varying slope regions provides an important tool in designing motion planning strategies for energy efficient walks in the quadruped. By applying control to get the quadruped towards the stable state path, the quadruped can be made to walk utilizing the natural dynamics and reducing energy consumption. Fig. 3.4 shows the Simulink model of a quadruped

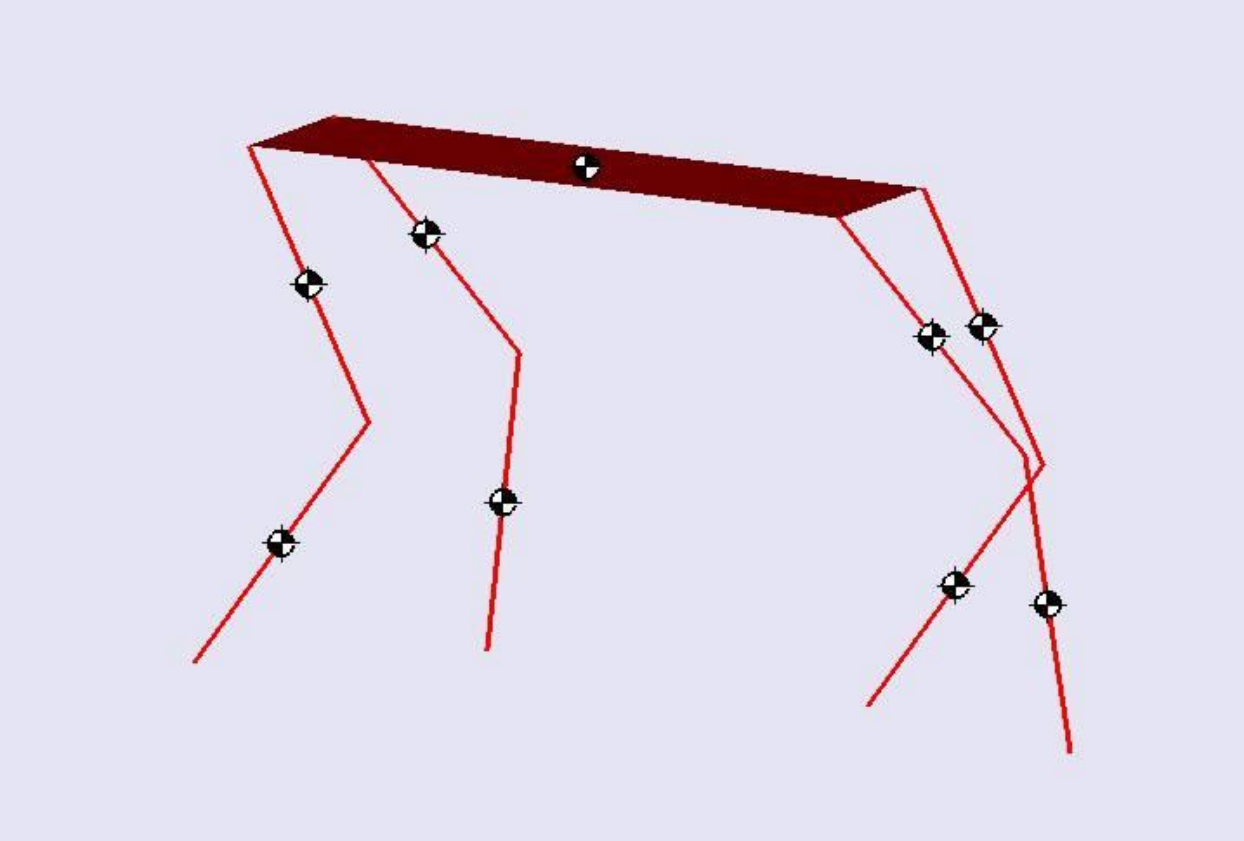


Figure 3.4 Simulink model of a quadruped

3.5 Simulink Block Diagram

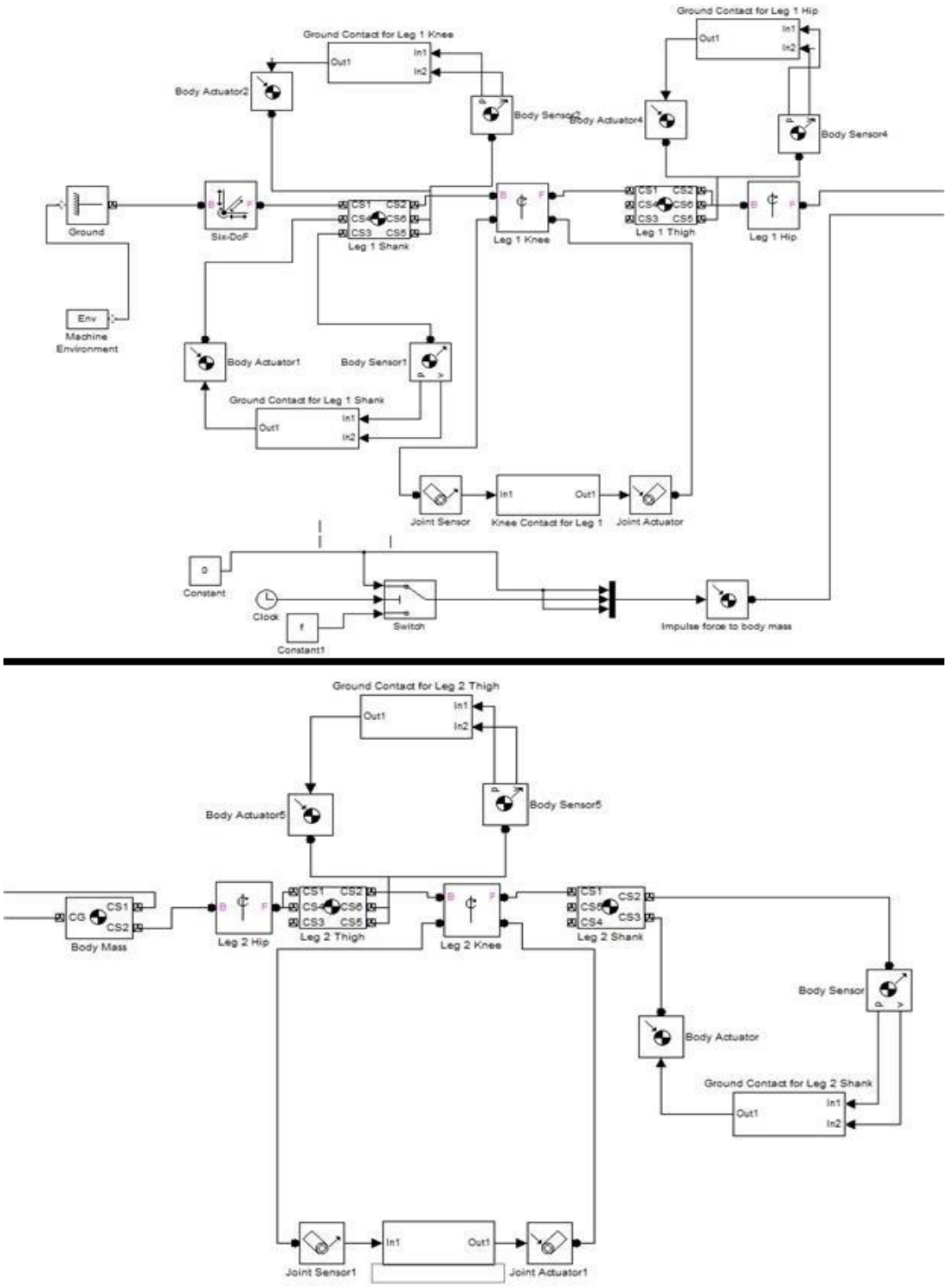


Figure 3.5 Simulink block diagram for the quadruped

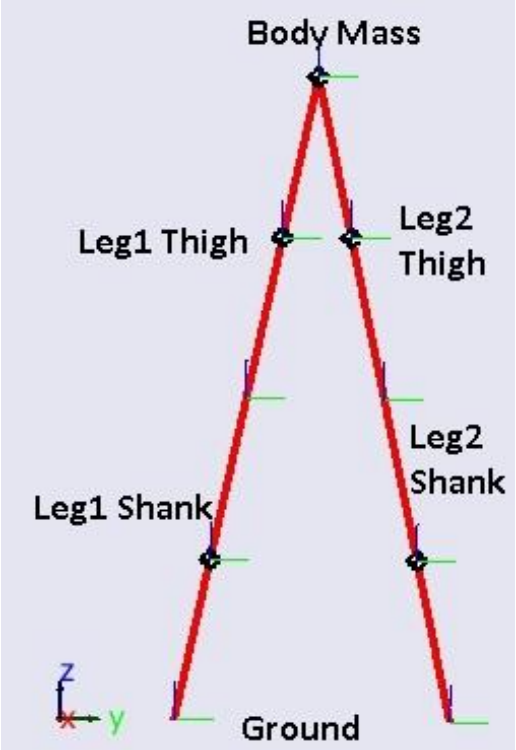


Figure 3.6 Simulink model of a Biped

The Simulink model of the bipedal walker and the initial configuration of the bipedal are shown. The explanations for some of the blocks used are provided below:

- **Machine Environment:**

  The machine environment port contains information about the gravity field the robot is kept it. The slope of the plane on which the robot is made to move is affects the gravity vector. The gravity vector is kept as [0 sin(gt) -cos(gt)]*9.81 where gt is the angle of inclination of the slope. By varying gt in a matlab function and running simulations, different passive walks can be attained.

- **Six DoF:**

  This joint block allows a unconstrained joint between the ground and the tip of the leg 1 foot.

- **Leg blocks:**

  Since in the bipedal there are two legs, the left leg or the initial swing leg is called leg 1 whereas the right leg or the initial stance leg is called leg 2. The shank is the part of the leg below the knee while the thigh is the part of the leg between the knee and the hip. The leg foot is the part of the leg that is supposed to touch the ground during stance position.

- **Body actuators and sensors:**

  These blocks allow the application of forces due to friction and contact. By sensing the contact between the ground and foot or during the stretching of the knee, an actuated PD (proportional and derivative) force is applied which is normally used to model hard stop conditions.

- **Ground Contact Sub system:**

  The ground contact subsystem contains the control force logic that models the hard stop condition when the foot touches the ground. This subsystem also contains the friction model of the system. The Simulink diagram of the subsystem can be seen below

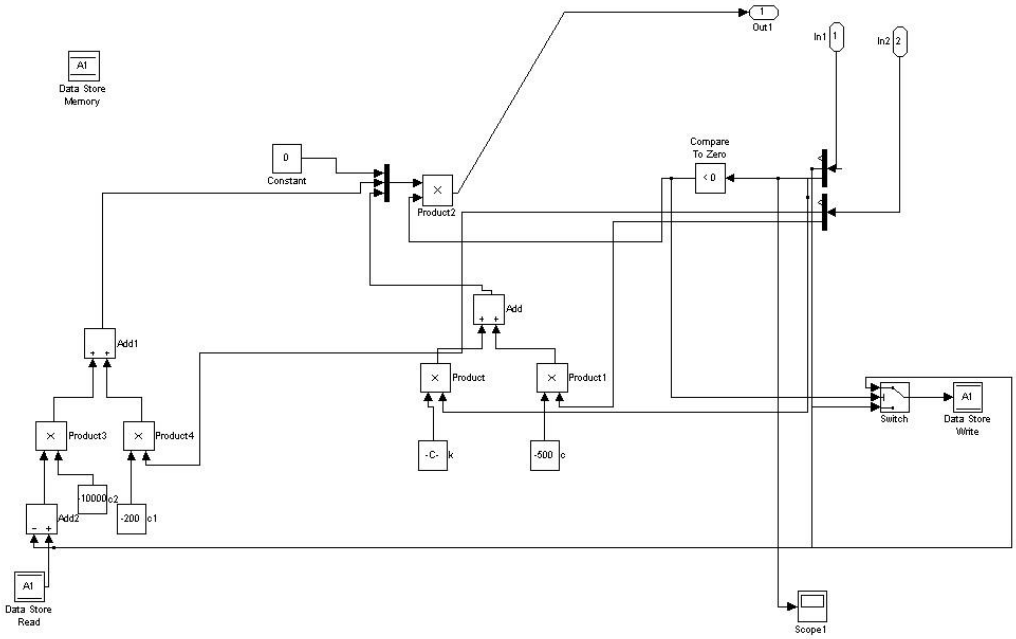


Figure 3.7 Ground Contact Subsystem

- **Knee contact subsystem**

  The knee contact subsystem models the force acting on the knee during hyper extension and makes sure to constrict the link during extension. The Simulink model of the knee subsystem is below.

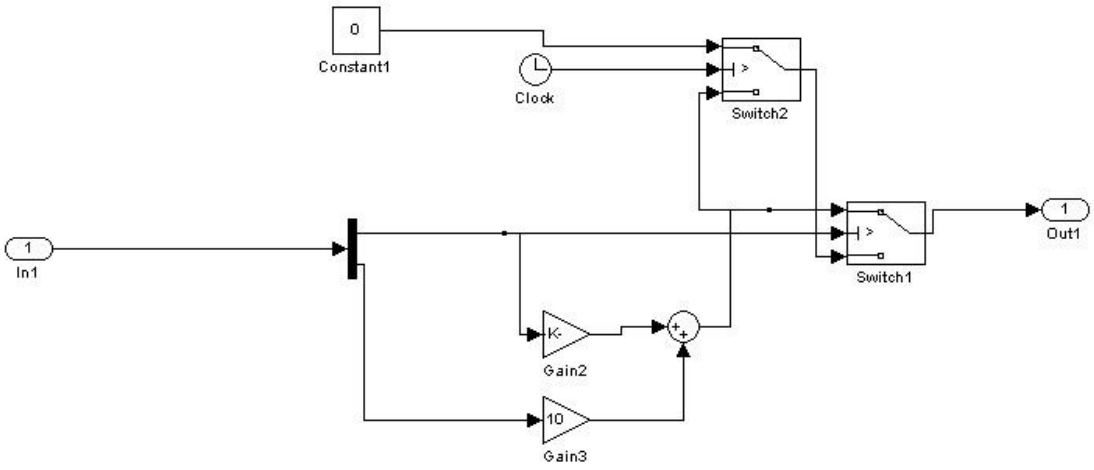


Figure 3.8 Knee contact subsystem

- **Body Mass**

  The body mass models the payload the robot is carrying. It also acts as a point where disturbances can be applied. The impulsive forces acting on the body mass are seen as the 'Impulse force to body mass' block.

3.6 Simulation Results:

In this paper, the simulations are carried out on a two legged kneed walker using all the methods described in (II). Simulations were carried out on the two legged walker as most of the work on passive dynamics have been performed on 2 legs. The Simulation based method of analysing passive dynamics is hence verified by comparing it with the existing methods. Fig. 3.9 shows the motion of the two legged kneed walker as simulated in Simulink.

Solving the passive dynamics for a similar kneed walker using methods similar to those previously proposed the following walk as seen in Fig. 3.9 is derived. The similarity of the gaits and movements derived by both methods can be readily seen.

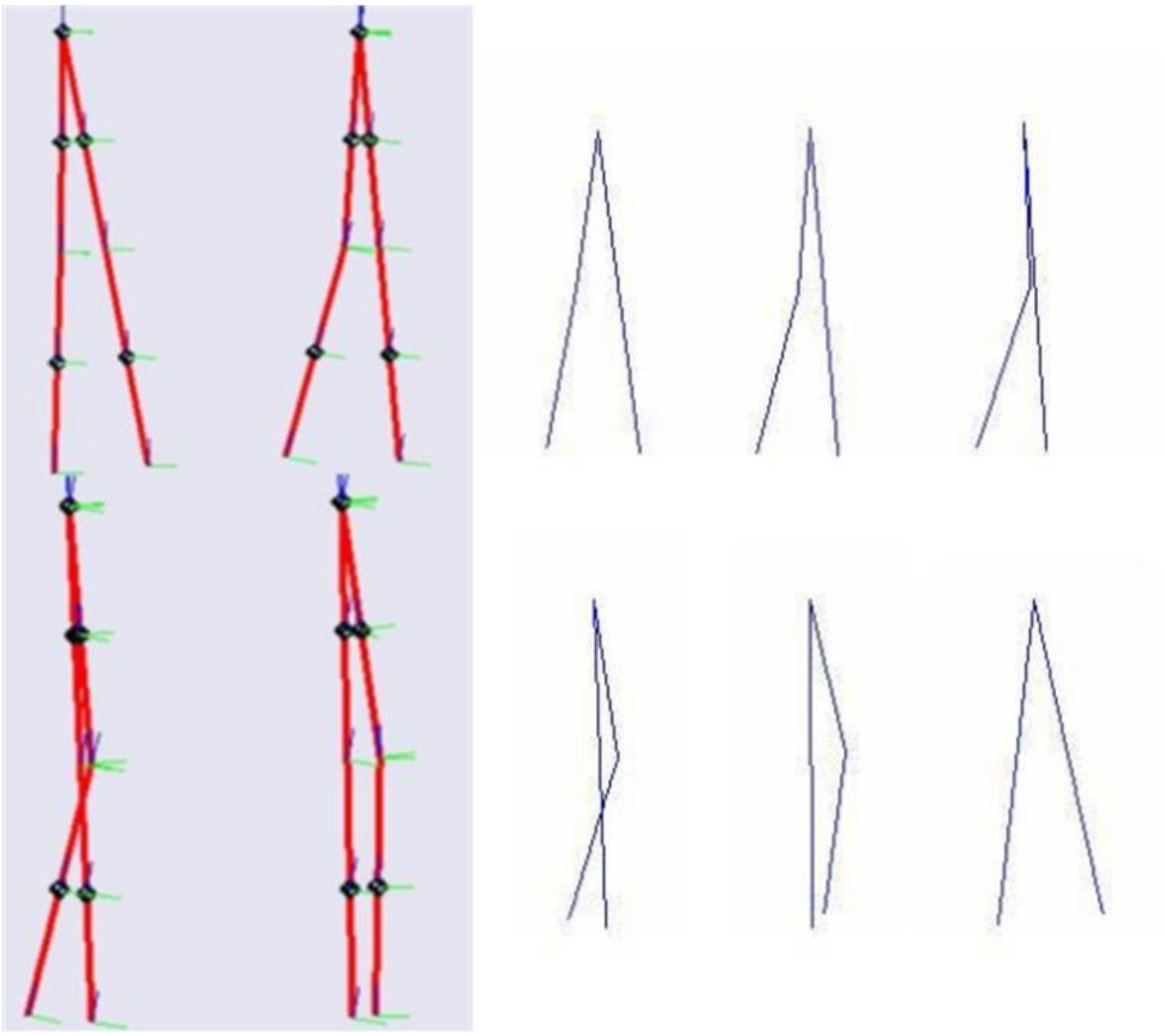


Figure 3.9 Passive walker animation on Simulink and the Kneed walker in Matlab

3.6.1 Perturbation Analysis in Kneed Walker

The stability of the walker was found by implementing a Poincare section after each foot collision with the ground. As a consequence, optimisation techniques such as Non-linear Least Squares and other algorithms were used in order to find the optimal configurations. At those optimal configurations, subsequent perturbations were given which made the system react in the manner given in the following figures.

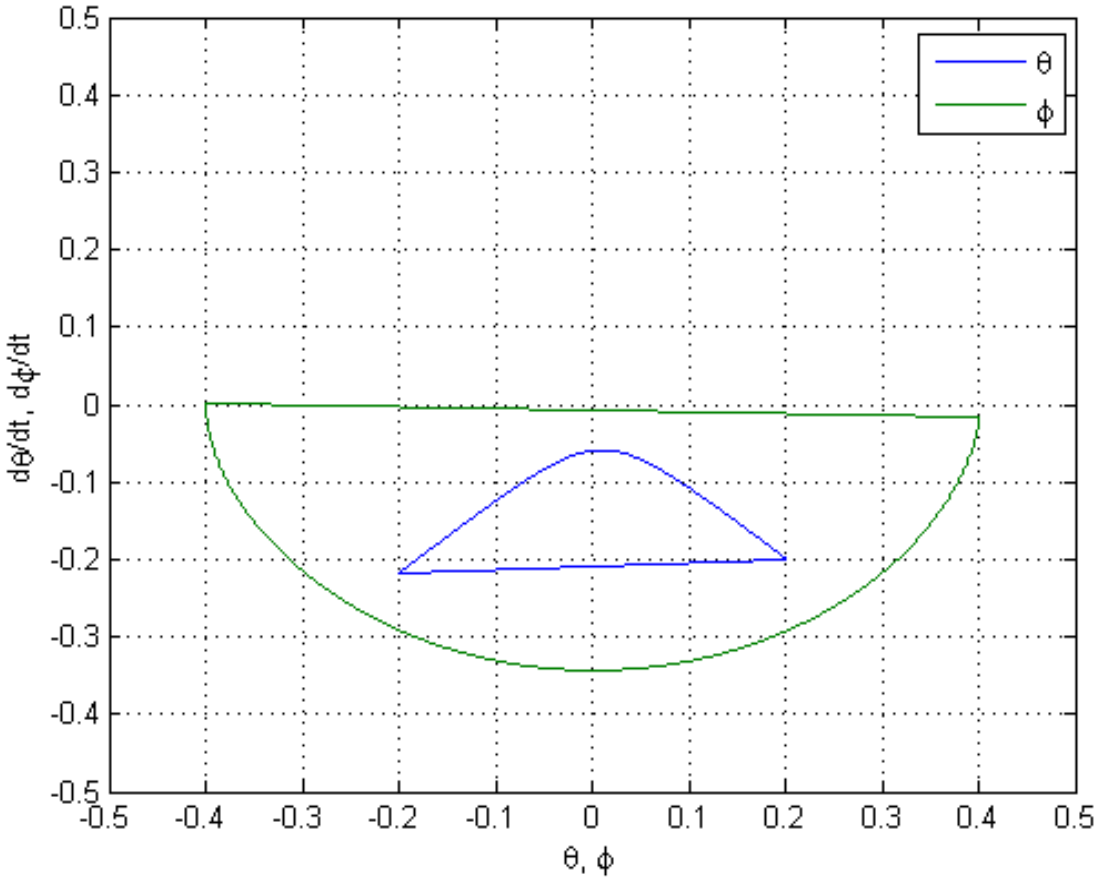


Figure 3.10 Limit Cycle in Kneed Walker

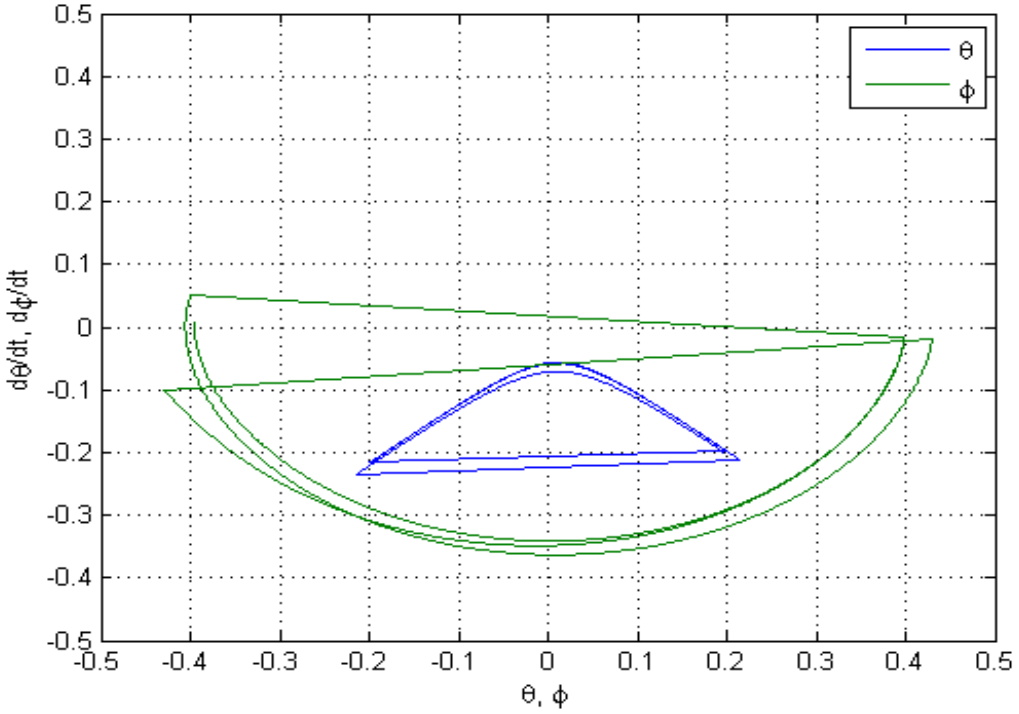


Figure 3.11 Limit Cycle stabilization after perturbation of 2e-4

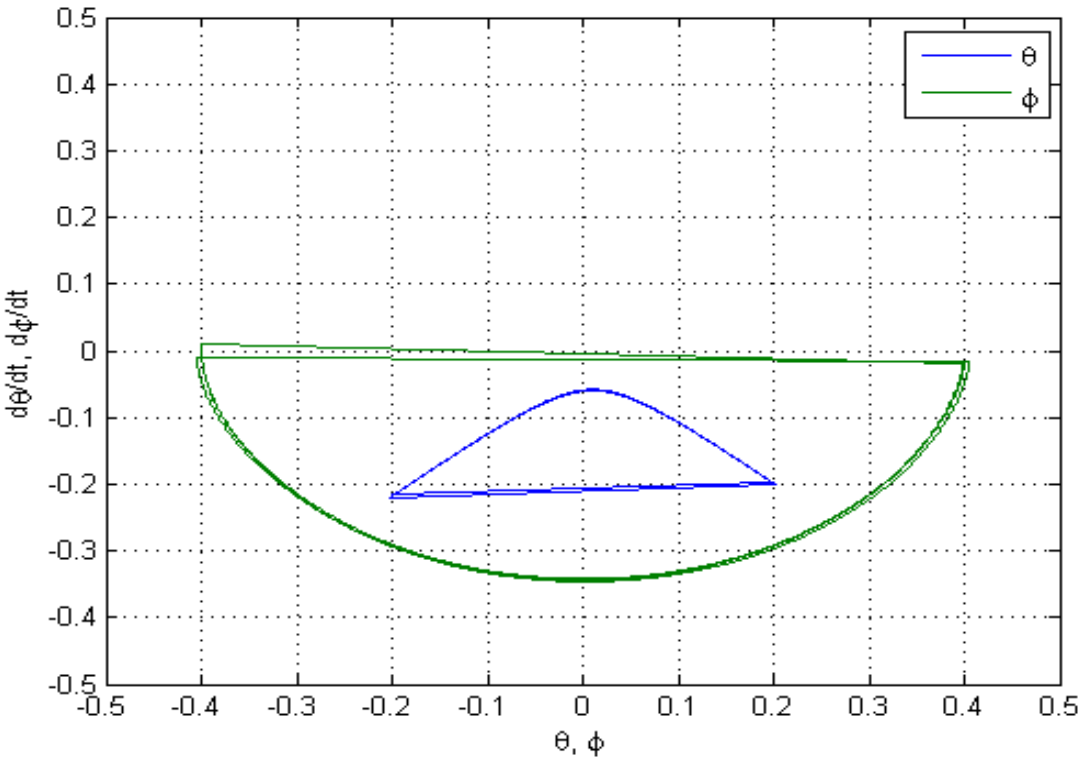


Figure 3.12 Limit Cycle stabilization after perturbation of 2e-5

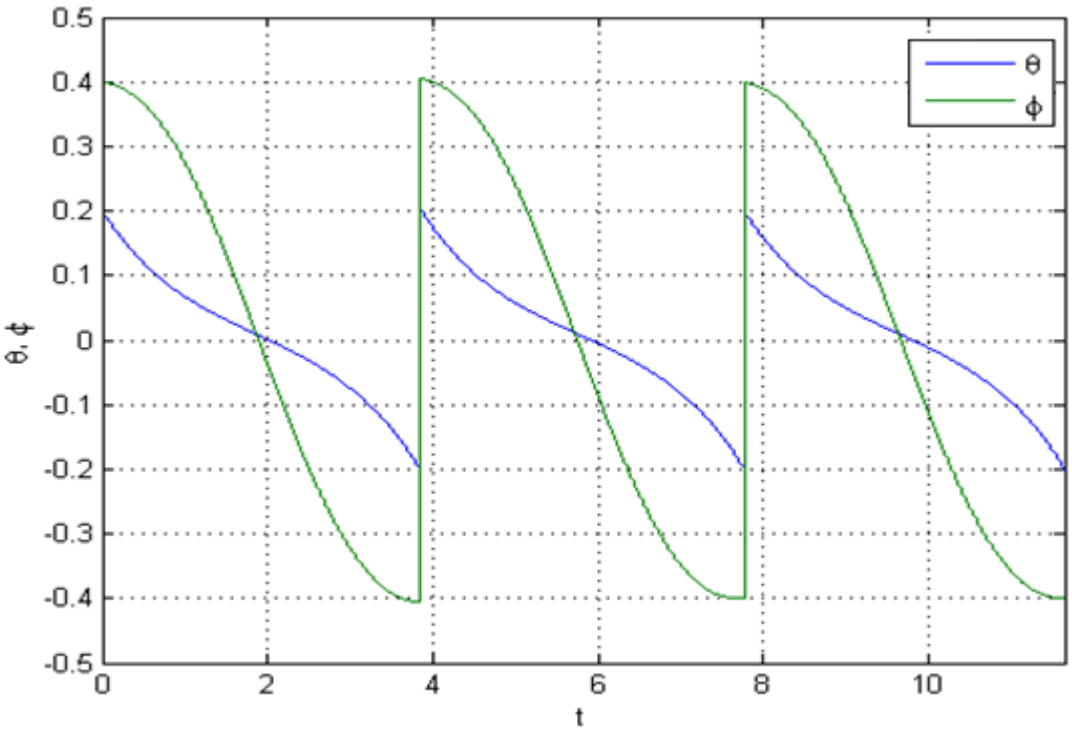


Figure 3.13 Periodic cycle of angles versus time

Chapter 4

DESIGN OF QUADRUPED

After having completed the literature review of existing quadrupeds and their functional characteristics like stability, passivity and motion planning, the next logical step in constructing a quadrupedal robot is to understand the physical problems associated with manufacturing a robot. Previous analysis that dealt with comparison between the kneed and quadrupedal models helped ascertain the similarity in functioning and therefore, simplified the simulation on Simulink. However, assembling a robot from scratch involves a much detailed process which would need to derive its rudimentary understanding from the literature review and further build upon it through a logical process. This step-wise demarcation of work needs to be addressed in a chronological fashion, in order to ensure that no steps are jumped or no logical conclusions are arrived at fallaciously.

Design being a tardy and tedious process would need to incorporate all possibilities and contingencies that must be pre-empted so as to remove all chances of error and make the robot as robust and stable as possible. This involves not just simulated analysis of the robot, but also producing CAD diagrams and models that help to further visualize the model before its actual assembly. CAD diagrams have been produced that rendered aid not only in the assembly of the robot, but have also provided a visual description of how the robot would appear if constructed, thereby allowing a possibility to make changes if necessary. Along with looking at the structural aspects, sufficient importance has also been given to monetary feasibility since the objective of constructing a robot would also be to realize that it is marketable and therefore, reasonably affordable. Hence, materials used and electronics applied have been scrutinized to see that expenditure is moderate so as to check the total cost of the robot. However, sufficient care has been taken to not be economical at the cost of reducing the performance of the robot. There is an existing trade-off between prices of each component with respect to their performance, and the most optimized function of both had to be selected, that could satisfactorily meet all the criteria.

The most appropriate procedure to be followed while attempting to design and develop a quadrupedal robot is as given in the following section –

4.1 Desired objective of design of robot

As has been sufficiently and elaborately elucidated in the earlier sections, the desired objective of developing a quadrupedal robot is to attain a system that has

- High stability –

  The robot should be able to remain standing in the absence of external forces and the gait should be designed so as to make the robot as stable as possible.

- Minimum energy expenditure –

  The placement of the motors on the legs should take into consideration the energy cost incurred by the motors in order to lift that load. Hence, strategic location and placing of motors which happen to be the heaviest component in the design of any robot, should ideally be positioned in order to reduce that expenditure.

- Reasonably affordable monetary cost –

  The most fundamental motivation is to ensure marketability which is what any design process should incorporate. The motors shouldn't be too expensive nor too cheap and therefore risk rendering low performance capabilities to the robot.

- Traversing capability –

  The primary motivation that may guide the development and design of a quadrupedal robot is its traversing capability. A typical quadrupedal robot is able to move forward and possibly backward if its design permits, along with permitting a yaw turn, that is move towards the left or the right.

- Degrees of freedom –

  Though not basic in its form, yet the degrees of freedom to be installed into the robot have to be higher since rotational freedom can only guarantee the achievement of the aforementioned objectives. To keep high complexity, the desired objective is to install 12 DOFs.

Though riddled with its high dimensional complexity, high energy cost and low stability, these desired objectives serve as the basic yardstick to identify and then further the design motivation of a quadrupedal robot. Moreover, since the dynamics of a quadrupedal robot are fairly complicated, the design will involve taking certain assumptive criteria so as to arrive at the structure of the robot.

4.2 Dynamics

Since the dynamics of a 12 degree of freedom (DOF) and their complexity have been reasonably talked about previously, the more convenient and appropriate approach to adopt is to understand the dynamics of a leg individually. For this purpose, the Lagrangian Equations of Motion can be used to arrive at the dynamics of a single leg.

To write the equations of motion, the Lagrangian, L, is defined as the difference between the kinetic and potential energy of the system. Thus,

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{4.1}$$

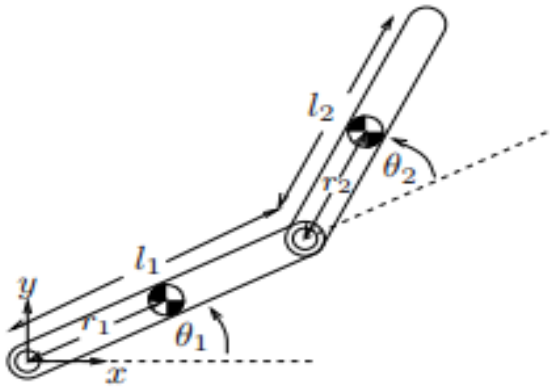


Figure 4.1 A two-link planar manipulator

And the Lagrangian equations are,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{dq_i} = \tau_i \tag{4.2}$$

Now, for a two-link planar robot, let r_1 and r_2 be the distance from the joints to the center of mass for each link as shown in the figure, where c_1 and s_1 correspond to $\cos(\theta_1)$ and $\sin(\theta_1)$,

$$x_1 = r_1 c_1 \tag{4.3}$$

$$\dot{x}_1 = -r_1 s_1 \dot{\theta}_1 \tag{4.4}$$

$$y_1 = r_1 s_1 \tag{4.5}$$

$$\dot{y}_1 = r_1 c_1 \dot{\theta}_1 \tag{4.6}$$

$$x_2 = l_1 c_1 + r_2 c_{12} \tag{4.7}$$

$$\dot{x}_2 = -(l_1 s_1 + r_2 s_{12})\,\dot{\theta}_1 - r_2 s_{12} \dot{\theta}_2 \tag{4.8}$$

$$y_2 = l_1 s_1 + r_2 s_{12} \tag{4.9}$$

$$\dot{y}_2 = (l_1 c_1 + r_2 c_{12})\,\dot{\theta}_1 + r_2 c_{12} \dot{\theta}_2 \tag{4.10}$$

Therefore, inserting these values into the Lagrangian equation,

$$\begin{bmatrix} \alpha + 2\beta c_2 & \delta + \beta c_2 \\ \delta + \beta c_2 & \delta \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -\beta s_2 \dot{\theta}_2 & -\beta s_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ \beta s_2 \dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{\tau}_1 \\ \dot{\tau}_2 \end{bmatrix} \tag{4.11}$$

where,

$$\alpha = I_{z1} + I_{z2} + m_1 {r_1}^2 + m_2 ({l_1}^2 + {r_1}^2) \tag{4.12}$$

$$\beta = m_2 l_1 r_2 \tag{4.13}$$

$$\delta = I_{z2} + m_2 {r_2}^2 \tag{4.14}$$

Using these values, the next step is to realize the design of the leg by placing sufficient masses and modelling the leg appropriately.

4.3 Design of leg

Once the dynamics of a 2-link planar manipulator have been understood, the mass of the leg along with that of the motor can be calculated, so as to arrive at the exact position of the center of mass of each leg. Using that, a premature design of the leg can be constructed.

It is first necessary to make a preliminary design of the leg itself. Here, the lengths of the links are as follows –

- Thigh (l_1) = 12cm
- Shank (l_2) = 14cm

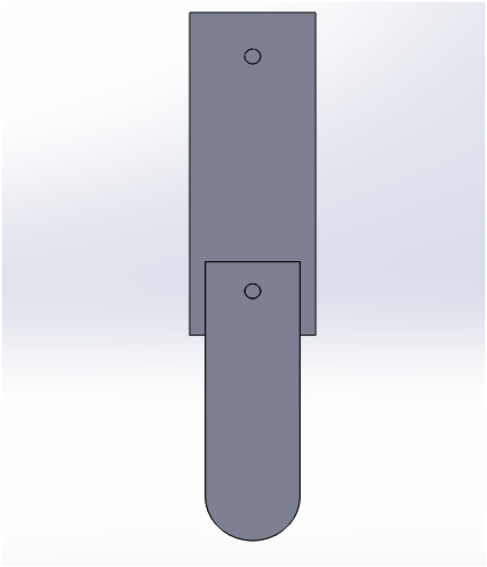


Figure 4.2 Premature design of the leg

The leg-length which is around 26cm has been chosen so as to give the optimum stride length with respect to the height of obstacles that it can traverse. However, a robot having legs longer would be prone to greater instability and buckling. This premature design allows one to arrive at the appropriate motors to be chosen to give the maximum advantage. To design the leg, it is crucial to finalize on the motors to be used because their weight will also contribute to the dynamics. The following figure lists the prices of the motors along with their weight and torque provided.

Table 4.1 Comparison Chart of motors

Motor	Torque	Weight	Price
1. NEMA23 Stepper Motor	19 kg-cm	1000 grams	Rs.4,600
2. High Torque Encoder DC Servo Motor	38 kg-cm	180 grams	Rs.3,950
3. High Torque Metal Gear Standard Servo	14 kg-cm	48 grams	Rs.1,050
4. 24V 100W DC Servo motor	5 kg-cm	1100 grams	Rs.2,800
5. NEMA17 Stepper Motor	4.2 kg-cm	350 grams	Rs.960
6. High Torque DC Geared Motor	12 kg-cm	180 grams	Rs.1,650

Therefore, after conducting a thorough survey of the motors in the market with respect to their characteristics such as weight, torque as well as with respect to price, the most appropriate motor which seemed to satisfy all these criteria is the High Torque Metal Gear Standard Servo. The primary reason to select this motor was due to its very less weight (48 grams) which can substantially help to reduce the overall weight of the robot, along with its high torque (14 kg-cm) that too at such low weight which gives the most optimum performance. The price is also considerably less (Rs.1,050) thereby making it economical too. The only drawback is that it is a servo motor, therefore it can only rotate in a discrete fashion (with step size of 1 degree) unlike the DC motor.

After finalizing the motor, the process of designing the leg of the quadruped can commence. The following are the detailed characteristics of the motor-

- Required Pulse: 3-5 Volt Peak to Peak Square Wave
- Operating Voltage: 4.8-6.0 Volts
- Operating Temperature Range: -10 to +60 Degree C
- Operating Speed (4.8V): 0.18sec/60 degrees at no load
- Operating Speed (6.0V): 0.14sec/60 degrees at no load
- Stall Torque (4.8V): 14kg/cm
- Stall Torque (6.0V): 16kg/cm
- Potentiometer Drive: Indirect Drive
- Bearing Type: Double Ball Bearing
- Gear Type: All Metal Gears
- Connector Wire Length: 12"
- Dimensions: 1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
- Weight: 48gm

The density of the acrylic sheet used to make the exterior of those links is about **1.2 g/cc**. Moreover, using this density, the weight of the links along with the motors can be found out –

- l_1 = 12cm X 2cm X 1cm
- l_2 = 14cm X 2cm X 1cm
- $m_1 = 2\ X\ \left(1.2\ X\ (\ 12\ X\ 2\ X\ 1)\right) + 48\ X\ 2 =\ 153.6\ grams$
- $m_2 = 2\ X\ \left(1.2\ X\ (\ 14\ X\ 2\ X\ 1)\right) =\ 77.2\ grams$

After evaluating the theoretical weight values, the motors are placed in such a manner so that the center of mass is concentrated on the thigh leg. This is done because unnecessary torque should not be exerted while trying to lift the motor when the shank link is being moved. It allows the center of mass to be higher (as opposed to if the motor were to be on the lower link), and therefore reduces unnecessary weight to be lifted by the motors. Therefore, common sense dictates that the motors for both the knee and the hip joint should be concentrated on the thigh if possible.

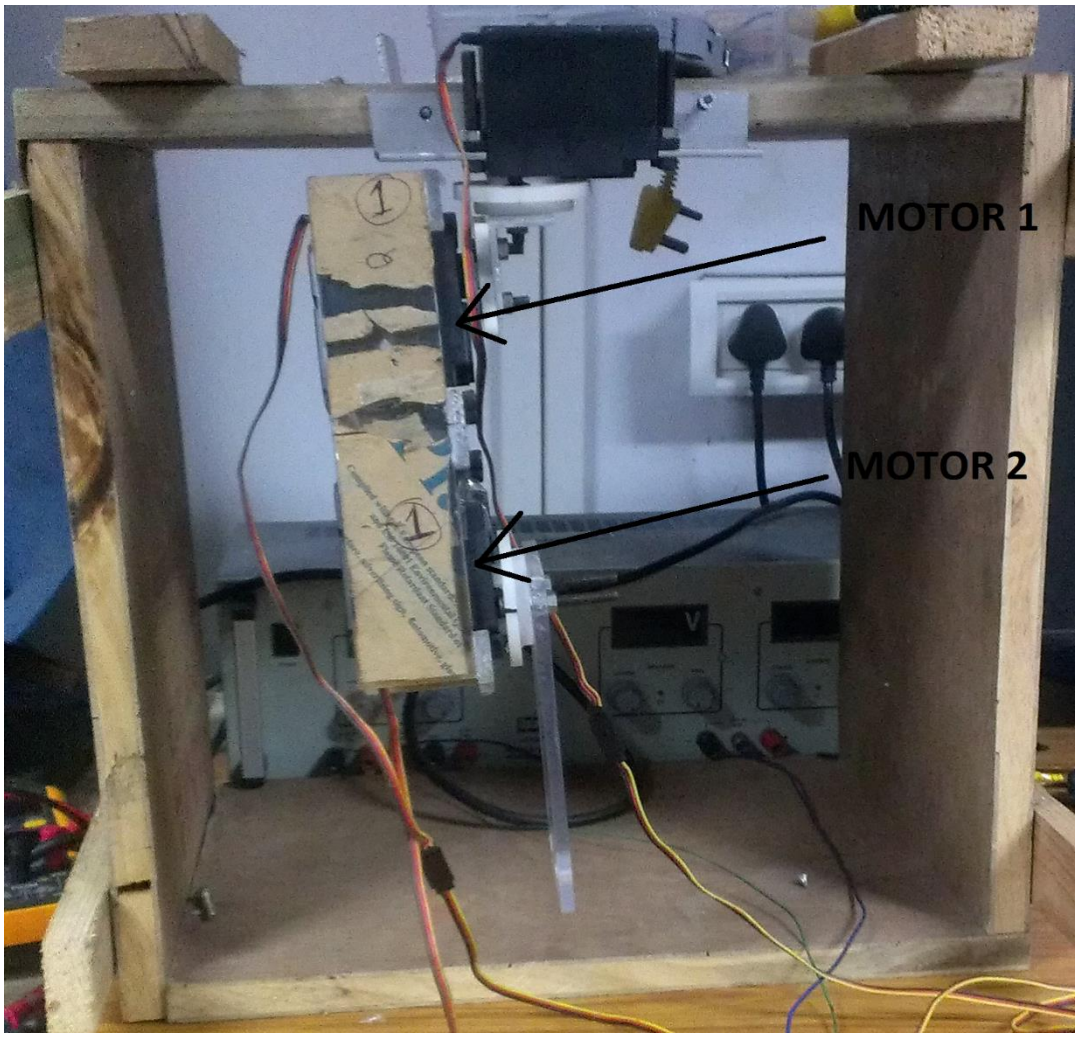


Figure 4.3 Motor placement in the leg

As can be seen in the above Fig 4.3, two of the motors have been placed on the thigh itself.

Moreover, the positioning of the motors approximately puts the center of mass of (l_1) to be approximately in the middle of (l_1).

Now that the length, masses as well as center of masses of the links have been fairly established, it is imperative to re-evaluate if the motors would be sufficient to drive the system or not. For this, two approaches have been primarily adopted.

4.3.1 Planar two-link analysis

It is important to understand if the hip motor would be able to sustain the weight of both the links along with the weight of the motors single-handedly. Which is why a proper analysis on its maximum torque is requisite.

- Maximum torque = 14 kg-cm
- Weight of the leg = 153.6 + 77.2 = 230.8 grams

The center of the mass of the leg is –

$$\frac{0.5\ X\ 12\ X\ (153.6) + (12 + 0.5\ X\ 14) X\ 77.2}{230.8} = 10.34 cm\ from\ the\ hip \quad (4.15)$$

Therefore, if the hip motor is to lift the whole weight of the leg, it would need a minimum of –

$$230.8\ X\ 10.34\ g - cm = 2.386\ kg - cm \quad (4.16)$$

Since the ratified value of the motor is 14kg-cm, it can very easily handle this required torque.

4.3.2 Full Body Analysis

Under this section, it is ascertained if the legs are able to sustain the load and the force exerted by the rest of the body when one leg is in the swing-phase.

For this purpose, it becomes necessary to consider the mass as well as the center of masses of the different links in this system including the legs as well as the body.

- Base dimensions = $31cm\ X\ 31cm\ X\ 0.5cm$
- Base weight = $1.2\ X\ (31\ X\ 31\ X\ 0.5) = 576.6\ grams$
- Leg length = $26cm$
- Leg weight = $230.8\ grams$
- Total body weight = $4\ X\ 230.8\ +\ 576.6\ =\ 1499.8\ grams$

It is important to realize here that this is only the theoretical body weight. Further mass would be added due to other materials such as screws, bolts, l-angles, etc. which at this moment are certainly extraneous considerations, which is why they've been kept outside the scope of this analysis.

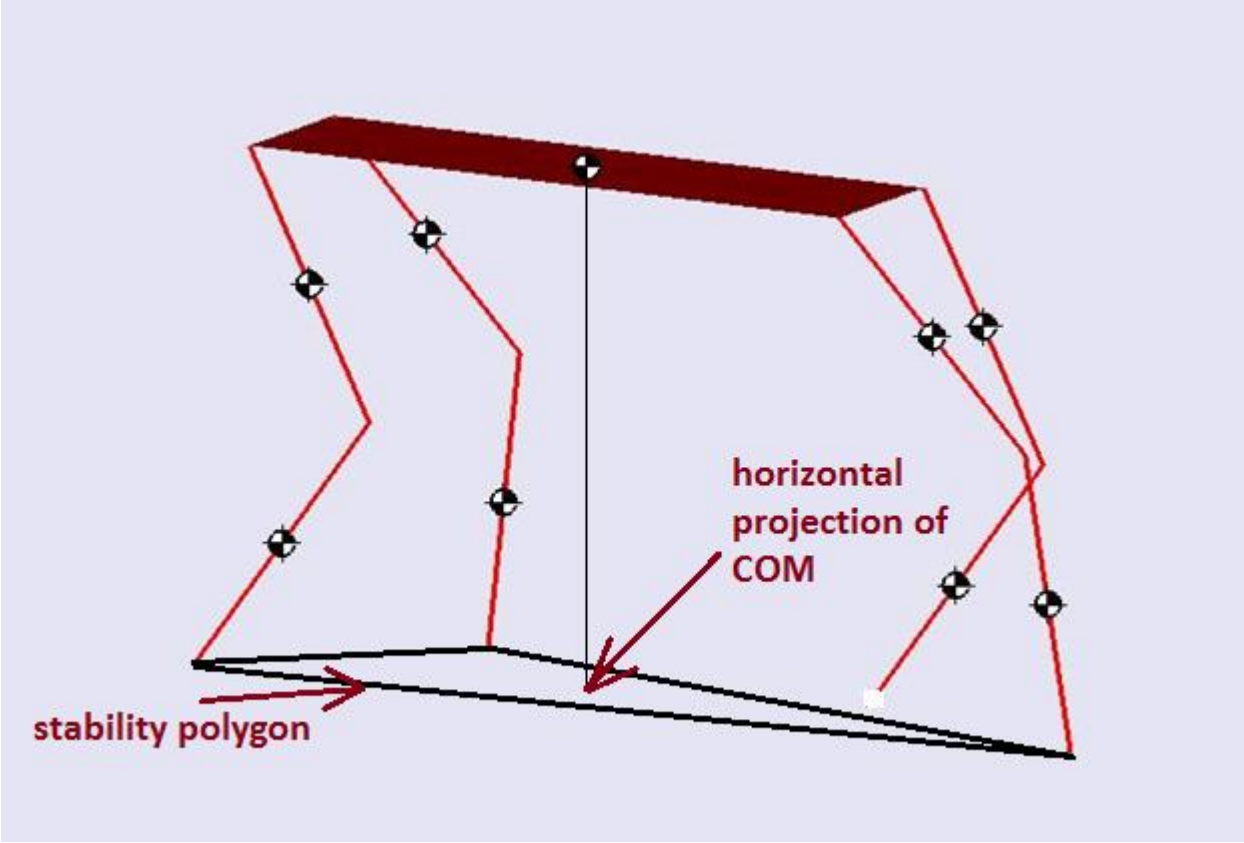


Figure 4.4 Stability polygon in walking gait

As Fig. 4.4 indicates, this stability analysis is to be conducted when one leg is in the air, i.e. in the swing-phase. The farthest that the Centre of Mass may move away from the stability region is when the swing leg is fully flexed outwards. In such a condition, the position of the center of mass and the required torque is to be analyzed in order to subdue this instability and prevent the robot from falling. For this purpose, a top-view approach of the robot is undertaken wherein the –horizontal projection of the center-of-mass would be assessed to view its degree of instability.

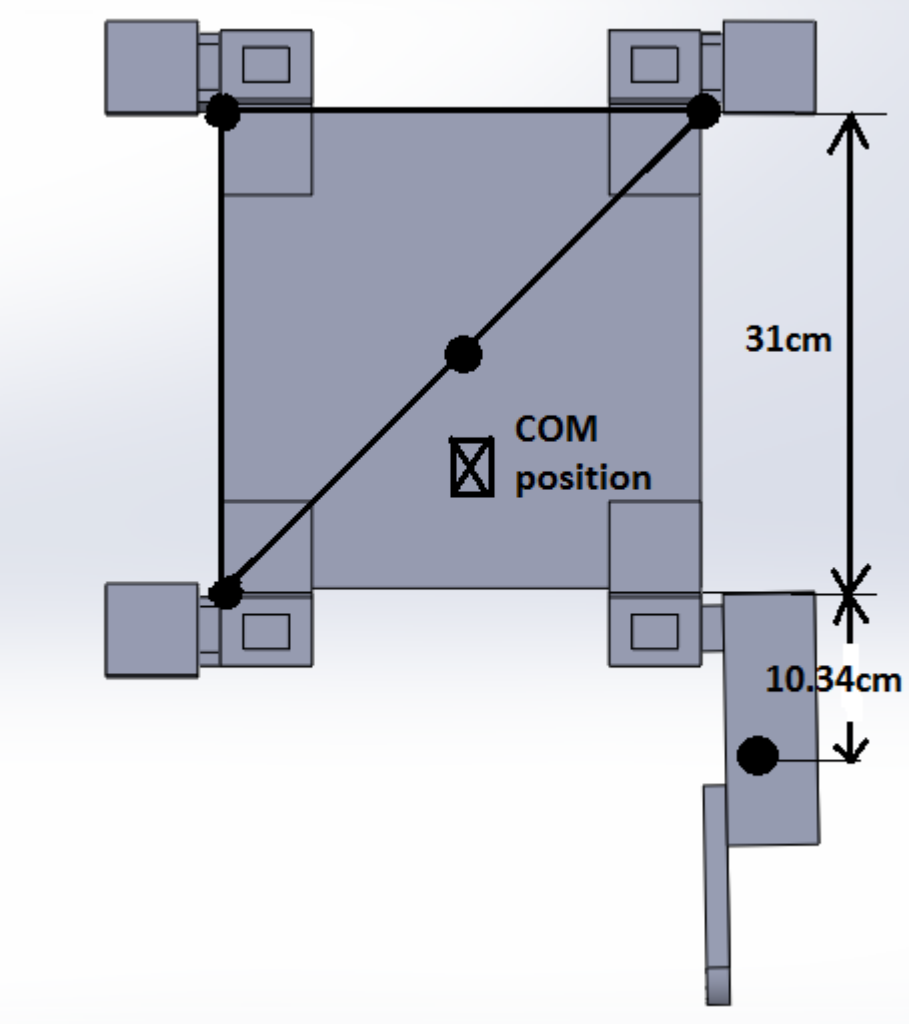


Figure 4.5 COM position outside stability polygon

As illustrated in Fig 4.5, when the leg is in its fully flexed position, the system would be at its most unstable point, thereby necessitating the largest torque from the motors. The distance of the COM (Center of Mass) of the complete body lies at a distance of 15.5cm along the y-axis since it is assumed that it would be in the center of its total length considering that all legs are symmetric. Therefore, calculating the COM position along the y-axis –

$$\frac{576.6\ X\ 15.5 + 230.8\ X\ 31 + 230.8\ X\ (31 + 10.34)}{1499.8} = \frac{25571.50}{1499.8} = 17.05\ cm \qquad (4.17)$$

It is important to realize in the above equation that the distance of 17.1cm is from the top-left corner along the y-axis as can be instinctively understood. Therefore, the distance or stability-margin would be –

$$17.05cm - 15.5cm = 1.55cm \qquad (4.17a)$$

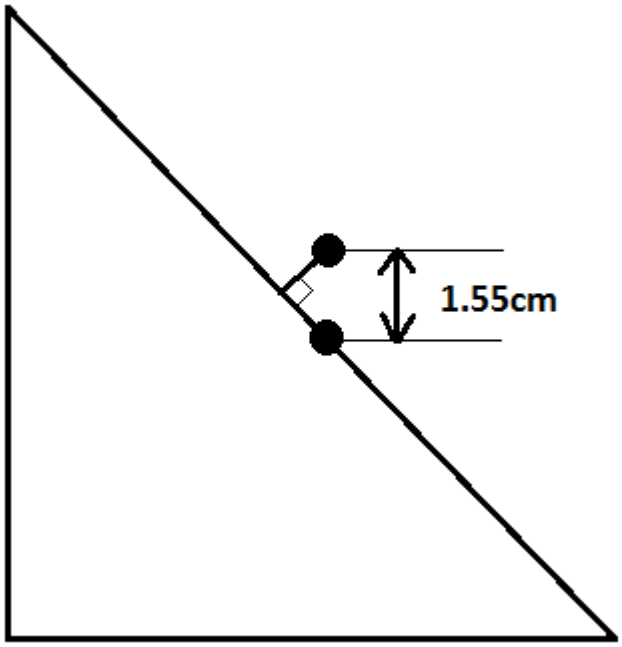


Figure 4.6 Stability margin

As shown in Fig 4.6, stability margin corresponds to the perpendicular distance between the center-of-mass position to the stability polygon. Here, the stability margin would be –

$$\frac{1.55}{\sqrt{2}} = 1.1\ cm \tag{4.18}$$

Therefore, the total torque required is –

$$1499.8\ grams\ X\ 1.1\ cm = 1.644\ kg - cm \tag{4.19}$$

Since the required torque is much lesser than any of the ratings possessed by the motors, therefore the static stability analysis holds true for the given motors as well.

4.3 Simulation of single leg gait analysis with preliminary design

After having assessed the motor's torque values, their placing and the value of link lengths, the next step is to simulate this system on a popular simulation software package such as Simulink in order to understand the analyzed gait that would be performed by the system. Moreover, simulation in a virtual physical environment allows one to test the designed system without actually building it, therefore giving the advantage of exposing any loopholes or drawbacks before the assembling process and also testing the gait on a virtual model that resembles the design objective.

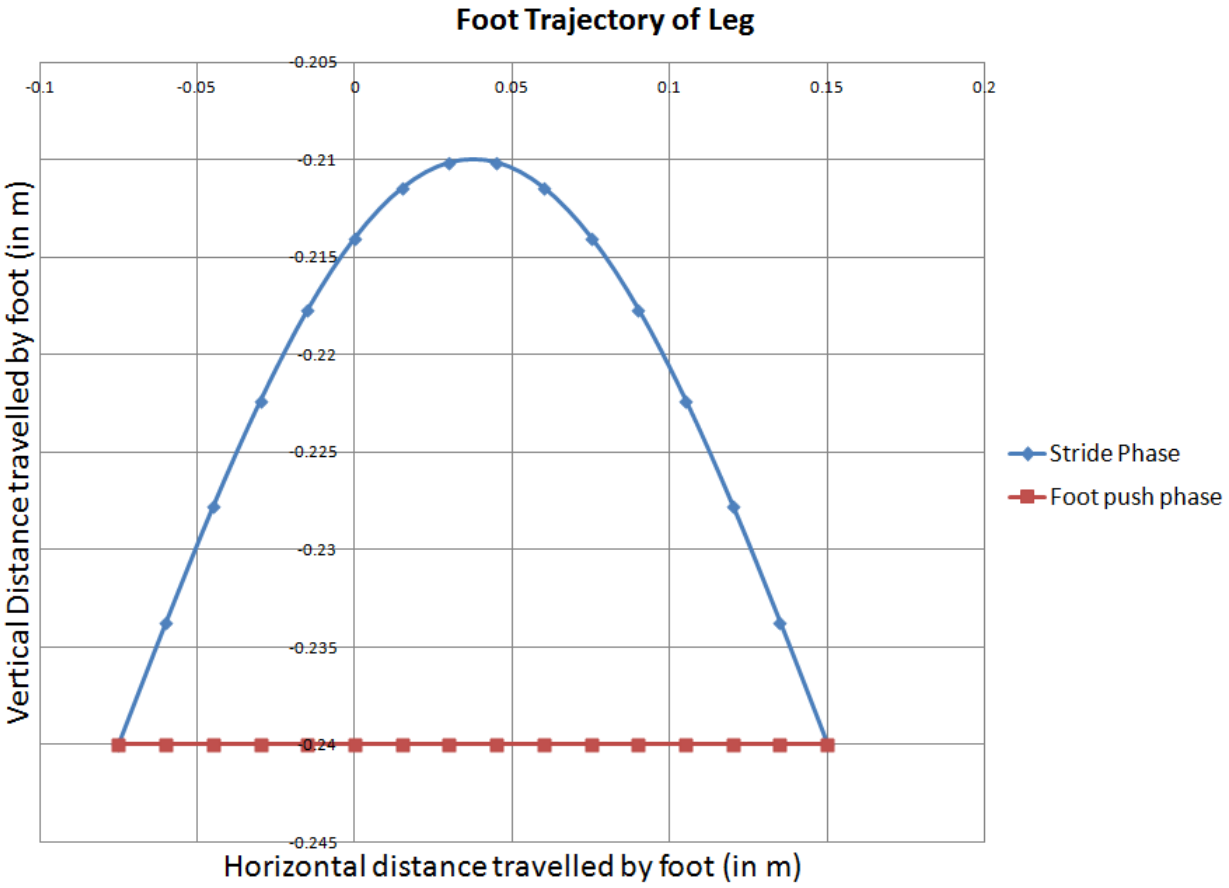


Figure 4.7 Foot Trajectory in the design gait

The gait used in the walking of one leg uses a foot trajectory as shown in Fig 4.7. This trajectory is essentially a scaled half sine wave during the stride. The stride length and stride height can be varied by altering the parameters of the sine wave. Moreover this variability of parameters allows for an onboard gait variation in the presence of obstacles. The foot trajectory chosen was taken to be sufficiently large with larger than usual stride lengths and stride heights. This was done to analyze the robot with a higher torque requirement than in practicality.

Before testing this gait on a full-body model, preliminary analysis has been done to see if the leg itself is able to follow the gait that has been provided to it. Here, the control strategy takes into consideration the requisite torque that needs to be applied by the motor in order to follow a pre-defined path that has been prescribed to it.

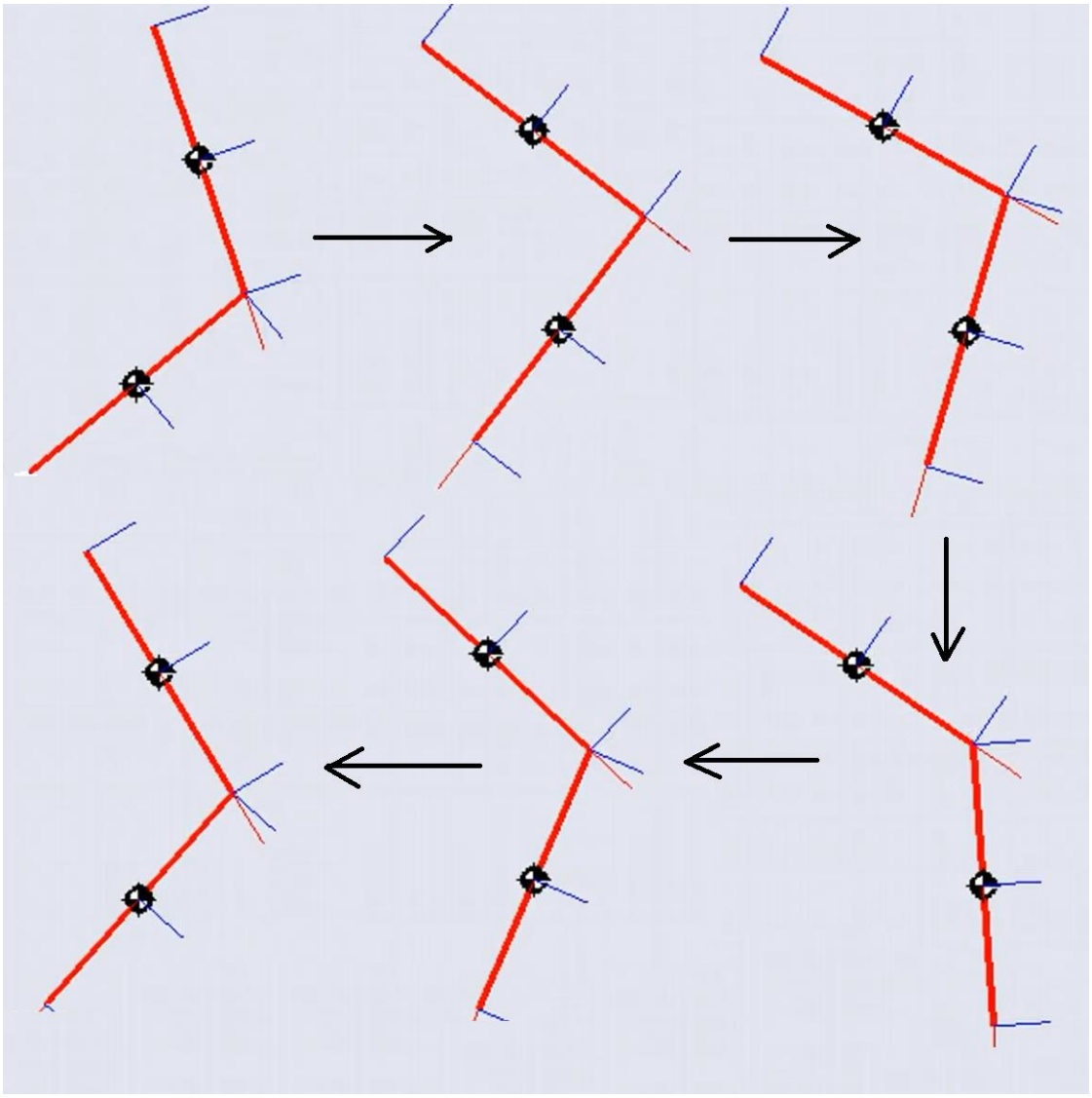


Figure 4.8 Two-link manipulator following gait using PD control strategy

The control strategy used was a trajectory following PD control on Matlab and Simulink (Appendix I). The physics engine in Simulink has the dynamic analysis in Section 4.2 inbuilt in its software, hence saving manual computational effort. Given the foot trajectory, using inverse kinematic relations, the required Hip angle and Knee angle trajectory are computed. Then a PD control generates the required torque to minimize the error between the traversed path and the required path. Since the PD control is applied via Simulink on a model of the leg using tunable controllers, the proportional and derivative factors of the control are tuned to obtain a response time on 0.1seconds. The result of the simulation is a generated leg gait as shown in Fig 4.8

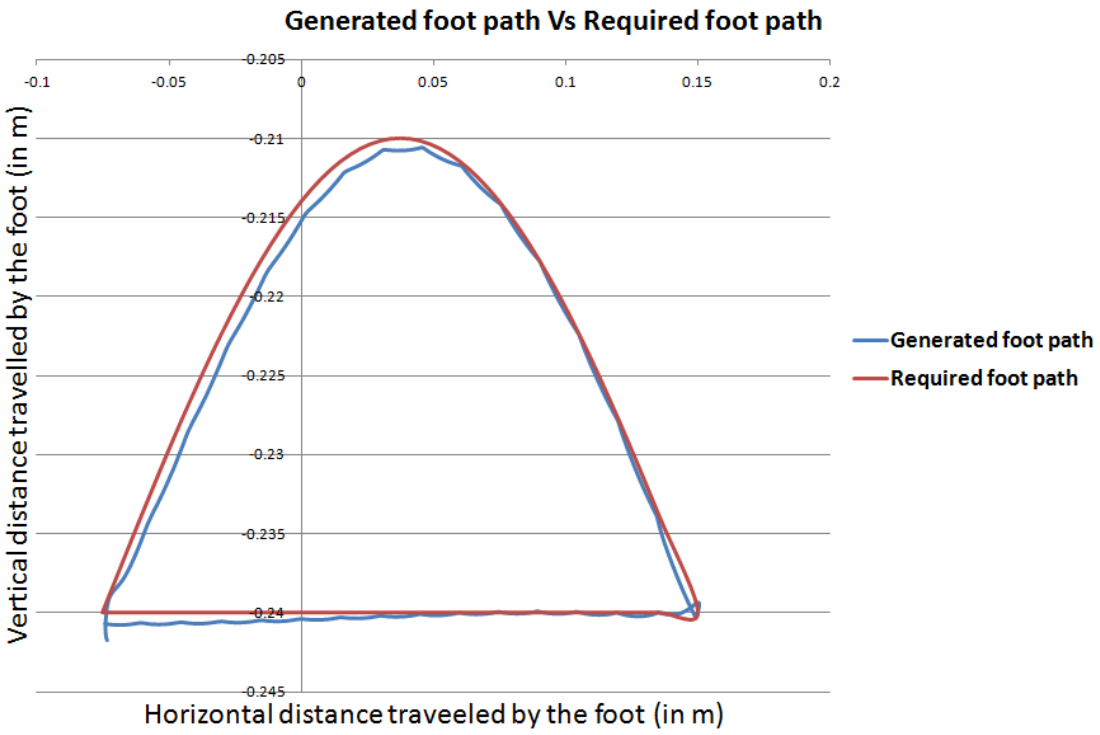


Figure 4.9 Comparison of control generated foot path and required foot path

Fig 4.9 shows the comparison of the generated foot path with the required foot path. Since most Servo motors offer a minimum response time of 0.1 seconds, the analysis shows that the leg and traverse the path without violating the conditions of maximum motor torque and response time.

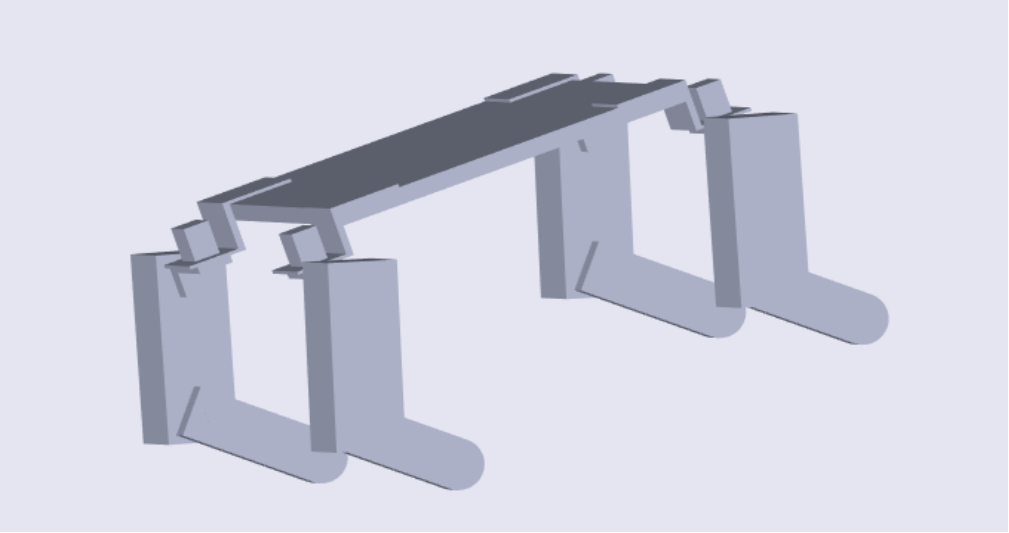


Figure 4.10 Simulation of model in Simulink

Fig 4.10 depicts a full-body simulation analysis using the same gait that has been applied previously on an individual leg. The model has been simulated in Simulink using the same link lengths, the center-of-mass of each of those links, along with the weight of the links and the central body frame that have been previously deliberated upon in Section 4.3.1.

- Thigh link length = 12cm
- Shank link length = 14cm
- Leg theoretical mass = 230.8 grams
- Body frame theoretical mass = 576.6 grams
- COM position of leg = 10.34 cm from the hip
- Total body weight = 1499.8 grams

The purpose is to ascertain if the values that have been accepted in the previous section can be applied on a virtual model with the same effectiveness or not. A natural walking gait was applied to see if the model had any existing instabilities or ineffective design which could potentially contribute to its failure. Since, the model walked with ease and there was no deviation in path observed, therefore it validates the values used for the model. Secondly, the simulation exercise also establishes the application of the gait on the model, which can be further used when assembly of the robot is complete.

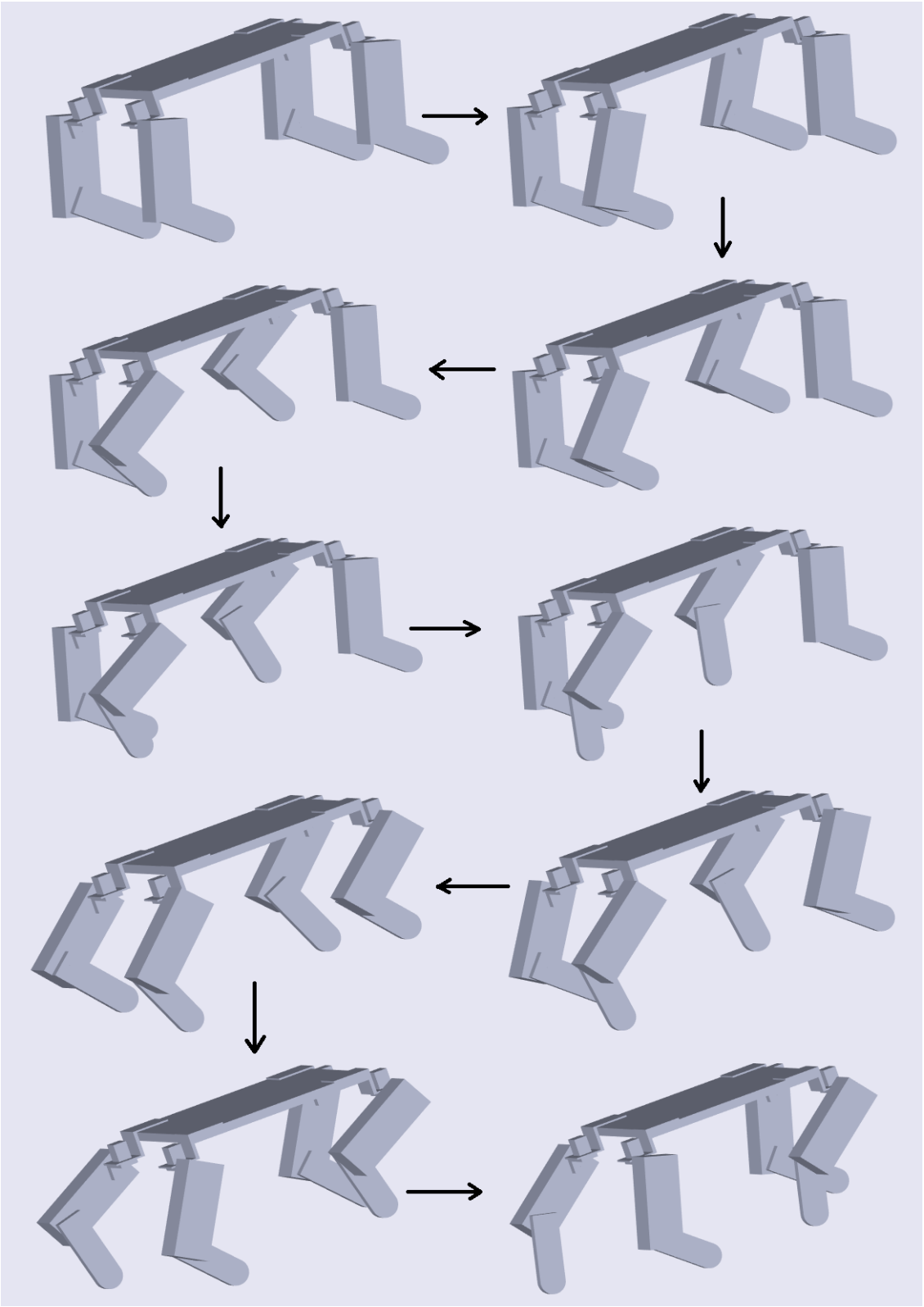


Figure 4.11 Application of walking gait in Simulink

The simulation, carried out on Simulink, generates a gait as seen in Fig 4.11. Complete codes can be found in Appendix II.

4.4 CAD model development

Previous sections have dealt with the dynamics, followed by a preliminary design and simulation that focused on validating that design. However, this section would be focused more on designing the model in a CAD software development package (SolidWorks) to finely and intricately model the various parts of the legs so as to configure the final design. In order to achieve that objective, a chronological approach is necessary.

4.4.1 Parts

First, the dimensions of the base have been assumed to be 31cm X 31cm (Fig 4.12). Since the base would be composed of an acrylic sheet, the thickness of the sheet used is 0.5cm.

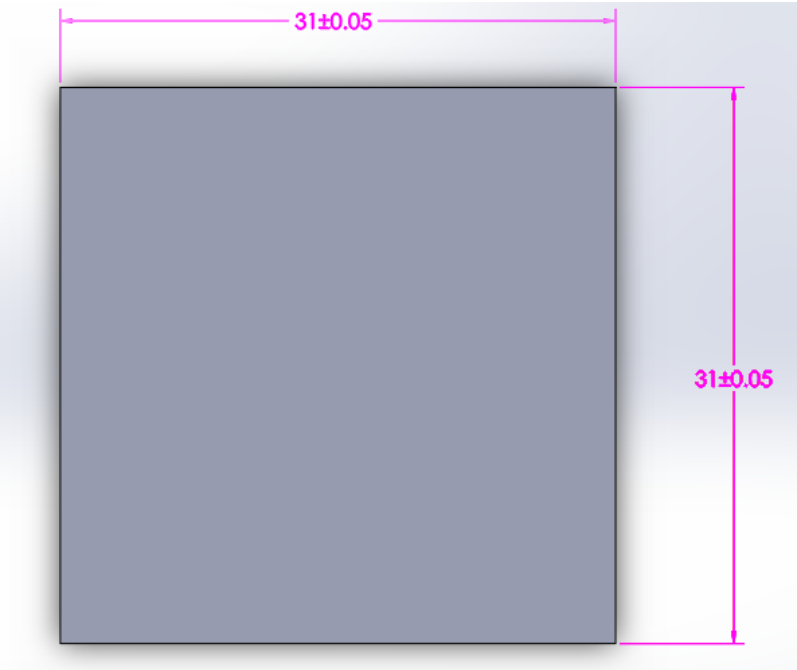


Figure 4.12 Base construction

The next component to design would be to construct the L-angles. The basic function of the L-angles is so that features that intend to be perpendicular to each other can remain so while also maintaining its sturdiness and rigidity. From the design considerations, it is also usually advisable while constructing a robot to make the L-angles as long as possible so that maximum holes can be drilled that would contribute to the sturdiness and reduce the vibrational aspects of the robot. However, the length should be an optimized value so that it would not exceed the monetary as well as weight considerations of the robot. Moreover, the L-angles to be made would be of Aluminum as it would significantly impart robustness.

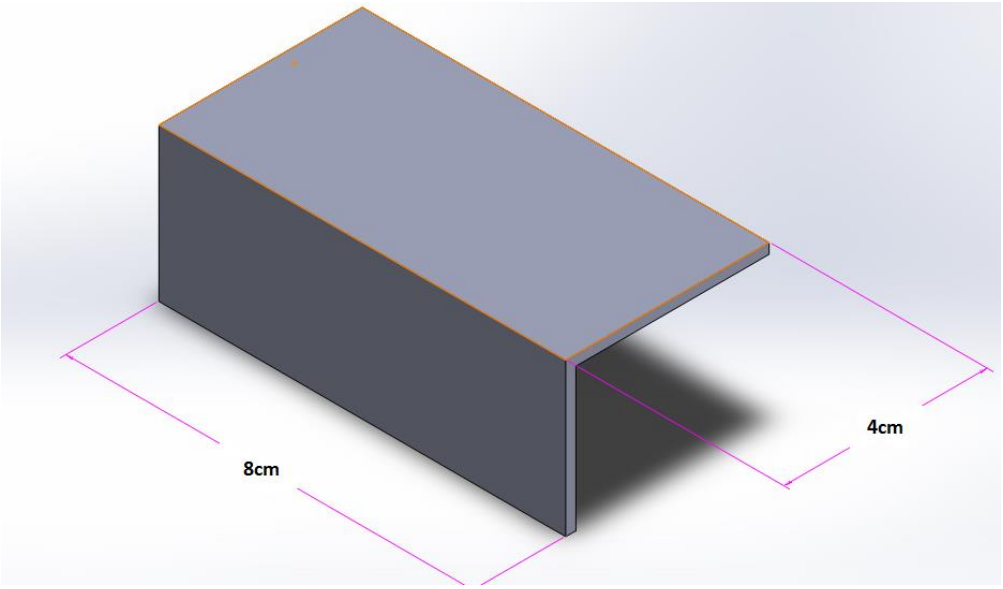


Figure 4.13 L-angle

As Fig 4.13 indicates, this L-angle would act as the connecting point between the body-frame of the robot with the legs. Subsequently, holes would be drilled in these L-angles to attach them firmly to the base. The dimensions of the L-angle are 8cm X 4cm. Fig 4.14 accounts for the yaw-motor that would enable the leg to rotate in the yaw-direction.

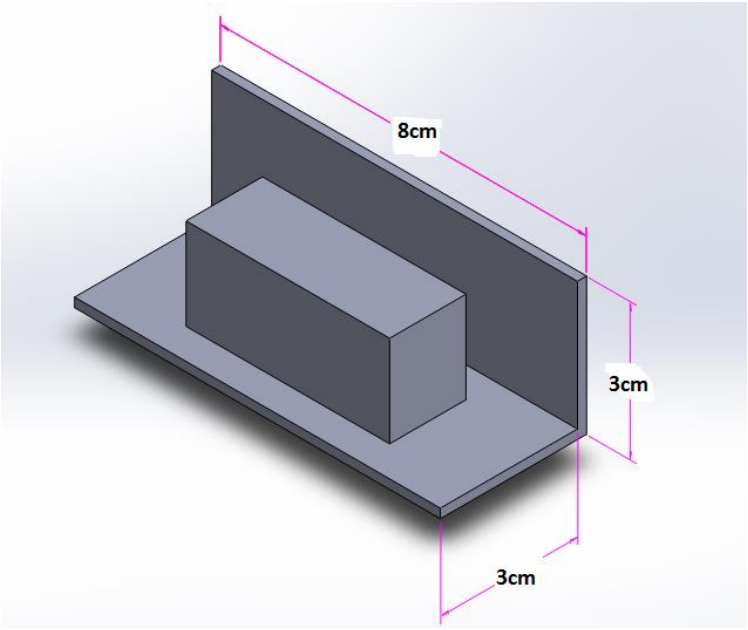


Figure 4.14 Yaw-motor design

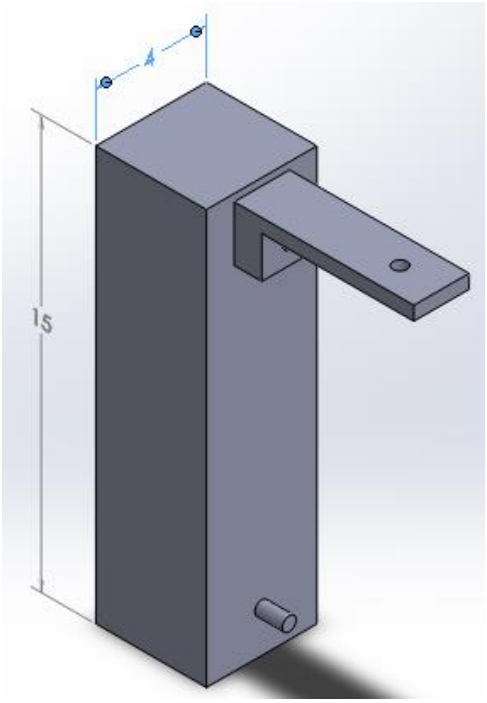


Figure 4.15 Upper link (thigh) design

Fig 4.15 depicts the upper link, i.e. the thigh of the leg and Fig. 4.16 depicts the lower shank link. The dimensions have been kept such that the linear distance between the joints amounts to 12cm whereas the total length of the link is 15cm.

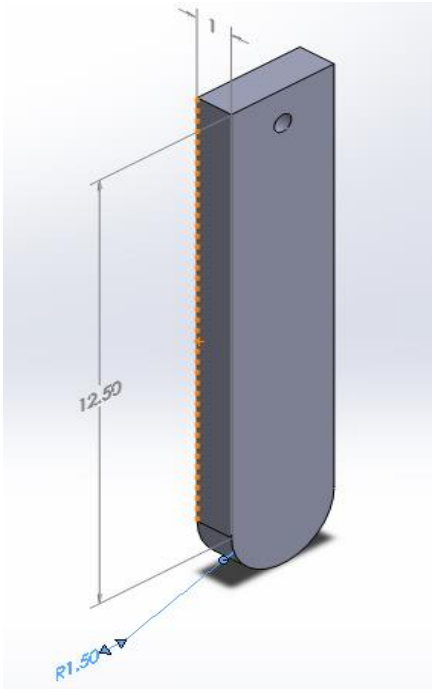


Figure 4.16 Lower link (shank)

4.4.2 Assembly

Assembling the different parts involves joining the thigh with the shank to form the leg, subsequently followed by joining all these four legs with the main chassis to build the quadrupedal robot

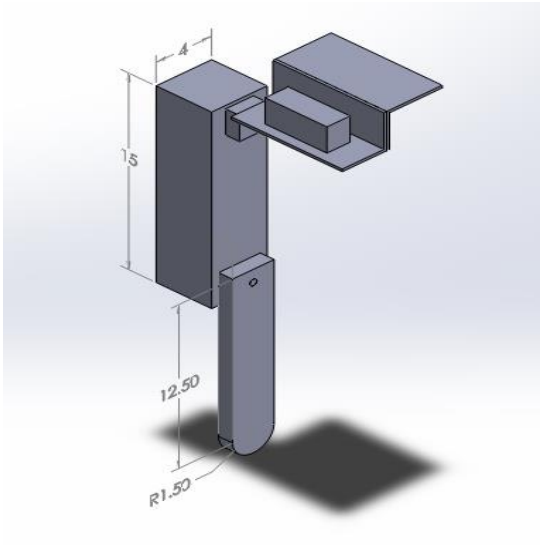


Figure 4.17 Leg assembly

Fig 4.17 shows the assembled leg after mating the individual components in SolidWorks, while Fig 4.18 shows the entire assembled quadruped.

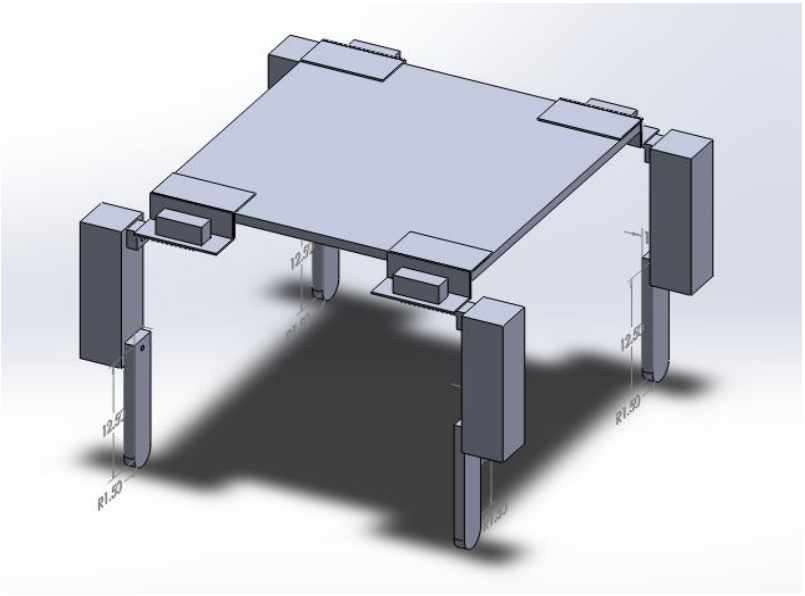


Figure 4.18 Full quadrupedal model assembly

4.5 Hardware Design

After conclusively establishing the design process along with the gait analysis and CAD model development, the next step is to actually build the robot. With the blueprint of the robot ready, the materials of the robot need to be procured. However, following a carefully calibrated approach is necessary otherwise it leads to unnecessary wastage of materials and subsequently, time.

For this reason, a test rig was installed to check if the performance of the motors is satisfactory and acceptable by normal standards. On this setup, the motor speed (rpm) was checked along with its highest rated torque to see if there are any prevalent errors surfacing within any of the motors. These motors were held together using a leg that was designed only for the purpose of testing the motors. The acrylic sheets were attached on both sides of the motors to give support. Moreover, the dimensions adopted were the same as in previous dynamic studies.

Figure 4.19 Test rig for individual legs

Following the tests conducted on this setup, the performance of these motors was satisfactorily proved. Thereafter, the process of building and constructing the robot needed to be initiated. An

acrylic sheet of 1m length was cut into differing lengths in order to develop the base of the robot. The primary frame was given a dimension of 31cm X 31cm in order to be compatible with the study on dynamics and gait done previously. Similarly, all dimensions of L-angles along with the link-lengths and motor placements were intended to be kept as close as possible to the values accepted in the previous theoretical tests conducted on scrutinizing the dynamics and the design.

After the conclusion of the development of the body, L-angles needed to be manufactured. For this purpose, aluminum sheets were cut to bring them to prescribed values. The L-angle dimensions were also slightly altered so as to allow more holes to be drilled.

Following the working on the L-angles, the legs were constructed with motors placed on the thigh link of the leg, thereby concentrating the mass of the leg upwards towards the hip. The dimensions of the legs constructed were the same as the one used for the test experiments.

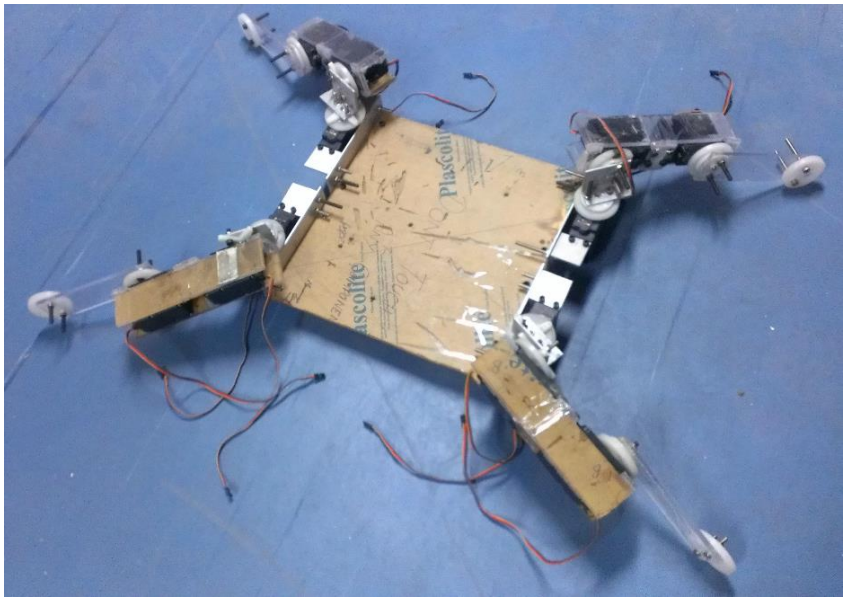


Figure 4.20 Robot assembly without electronics

After the basic hardware elements had been incorporated, it was imperative to include the electronics, i.e. the microcontroller along with its requisite motor drivers and wire connections. The Servo motors installed on the legs of the quadrupedal robot require 6V of voltage supply at 1A. For this purpose, for the given power drawn, a Servo Shield electronic circuit is used. The Shield circuit is mounted on an Arduino Mega 2560 controller board. This board has 256 KB of Flash Memory with a clock speed of 16MHz. Also with 54 digital IO pins this controller board

allows further expansion of the quadruped. The power is supplied externally through a variable voltage power supply (0-15V) at 30A.

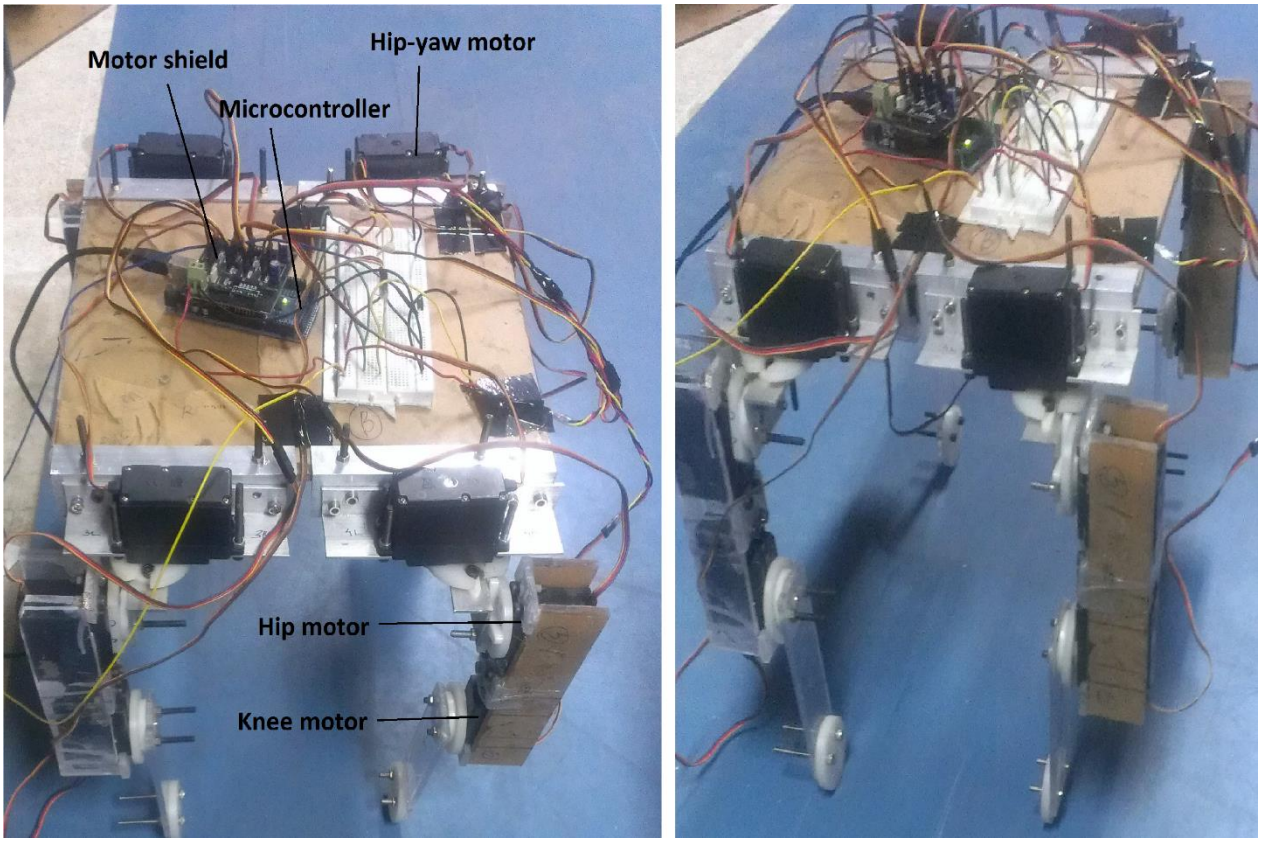


Figure 4.21 Robot construction with complete electronics

4.6 Method of Working

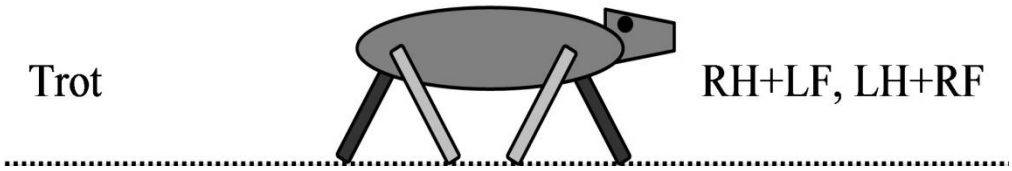


Figure 4.22 Representation of the Trot gait (courtesy www.intech.com)

The gait style implemented is the trot gait (Fig 4.22). The trot gait as mentioned in section 1.6, the trot gait involves moving the Right Hind leg (RH) and Left Fore leg (LF) in an identical manner. Similarly the Left Hind leg (LH) moves identically with the Right Fore leg (RF). When the RH+LF pair starts its stride phase (Fig 4.7), the LH+RF pair starts its push phase. Since the entire motion

is synchronized, when RH+LF finishes its stride, LH+RF finishes its push. Now the half cycle repeats with RH+LF now starting with the push phase and LH+RF starting with its stride phase. The quadruped hence moves in front through this alternating stride and push phases by its identical pairs. This manner of walking although statically unstable, allows the robot to move at a faster pace in comparison with the statically stable walk.

For the implementation of this manner of walking, first the joint angles derived in section 4.3 are stored on the Arduino Mega 2560 controller board. Then the controller generates the required individual motor angles based on the single leg gait and the trot gait style. Since the Servo motors installed have an inbuilt feedback controller (in this case a PD controller), the angles can be controlled by transferring the data of required motor angles to the respective servo motors. The motors move in the manner dictated by the angles generated in the Arduino controller and the trotting gait is generated on the physical robotic quadruped.

Chapter 5

EXPERIMENTS

The work primarily done till now entails the study of robot dynamics, gait analysis, designing and hardware construction. In this section, experiments have been conducted on essentially three different robots.

5.1 Quadrupedal Test Model

The paradigm of testing the gaits or the walking patterns of any quadrupedal robot necessarily involves a lot of risk since ineffective or faulty gaits can cause great damage to the machine. The fault may either be due to incorrect representation of the trajectory of leg angles with time or some inherent instability entrenched by the executed gait. Hence, there is a need to acquire a testing machine where one may be able to execute commands and gaits in a real-world physical environment and not worry about the damage involved. For this purpose, a ready-made robot chassis has been used, as shown in Fig 5.1. The additions made to this chassis are:

- Arduino Uno controller: This controller is smaller, less powerful version of the Arduino Mega 2560.
- Servo Shield: This circuit board is similar to the one on the larger quadruped (section 4.5)
- Foot grip: The grip on the foot was changed during experimentation due to the slipping of the previously existing feet on smooth grounds.

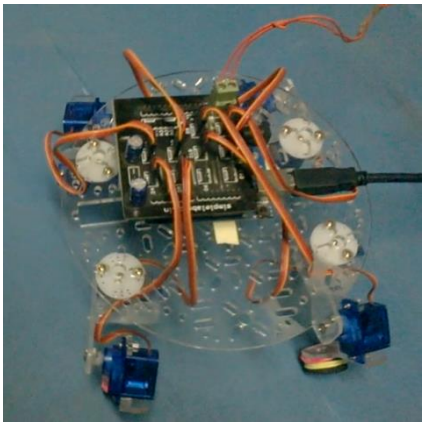


Figure 5.1 Quadrupedal Test System

In the crab gait, the legs move essentially in the plane of the robot body. Akin to crabs, the legs should be able to rotate in a clockwise or anti-clockwise direction as opposed to the conventional manner in which legs move forward in order to complete the gait. However the crab gait is very similar to the trot gait because both these gaits involve keeping two legs in contact with the ground while raising the other two in the air so as to actuate the robot, and therefore it enables one to make a reasonable comparison. The legs follow the movement as shown in Fig 5.2. The red lines connect diagonal feet while the green line depicts the leg which has lifted itself for movement.

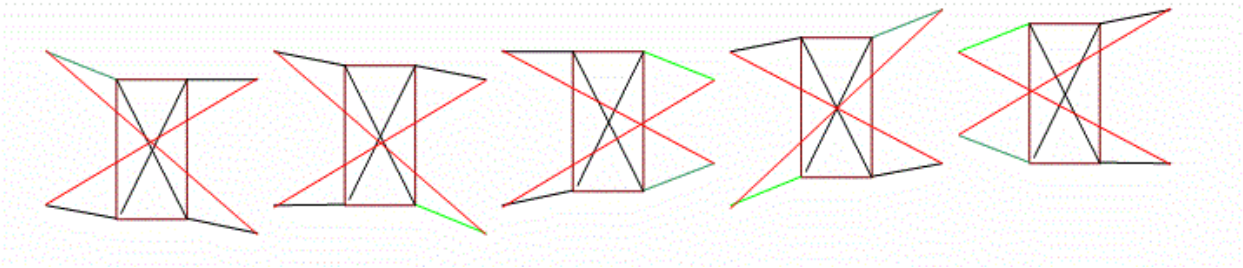


Figure 5.2 Crab gait in 4-legged robots (courtesy www.lynxmotion.com)

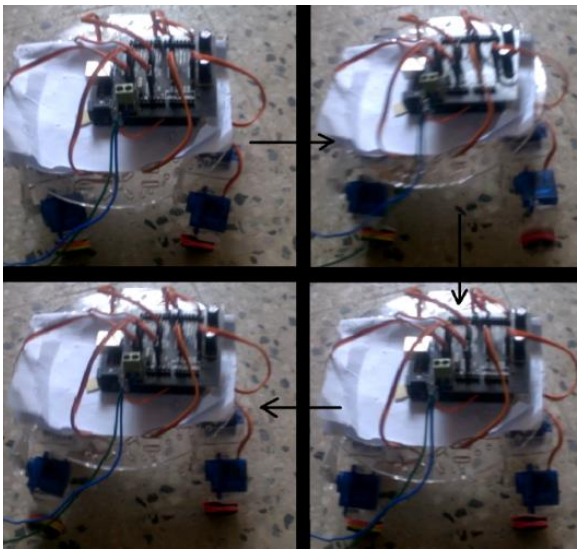


Figure 5.3 Crab gait on the test model

The manner of implementation of the crab gaits (Fig 5.3) is similar to the trot gait implementation discussed in section 4.6. The following crab gaits were implemented on this robot:

- Forward walking: The test robot moves straight ahead with little or no change in heading direction.

- Backward walking: The test robot moves backward in the reverse manner of the forward walking
- On the spot turning: The test robot turns at its spot with moving its Center of Mass (COM) to a large extent. This type of turning is also called zero radius turning
- Simultaneous walking and turning: By altering the step lengths of the feet, it was seen that the test robot turns as well as moves it COM to a large extent. This type of non-zero radius of turning allows maneuvering a turn more easily.

5.2 Quadrupedal Model

Since the basic quadrupedal model has been designed, analyzed and subsequently developed in the previous sections, with subsequent testing being done on a test quadruped as well, it is now permissible for those same gaits to be applied on the quadruped. As opposed to the crab gait that was executed on the test robot, the trot gait would be applied on the quadruped to test its stability and observe its walking cycle. However, they would be clubbed under two different experiments – one where a forward-walking gait is implemented, and another where a turning gait is incorporated.

5.2.1 Forward-walking trot gait

Fig 5.4 explains the hip and knee-angle in a forward walking gait which are important in understanding the forward-walking gait.

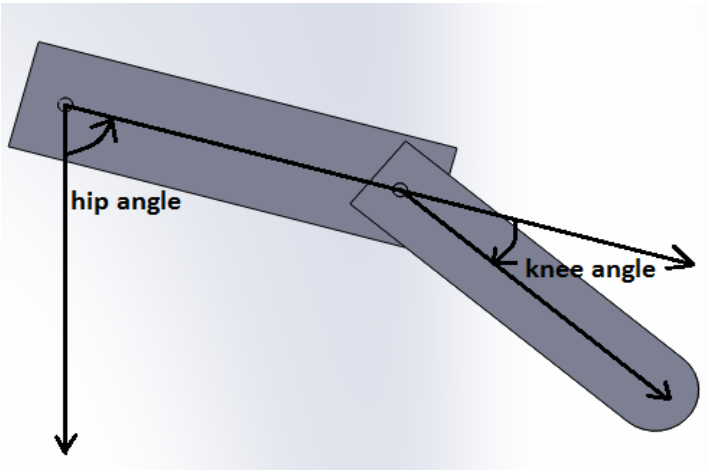


Figure 5.4 Hip and knee angle in forward-gait

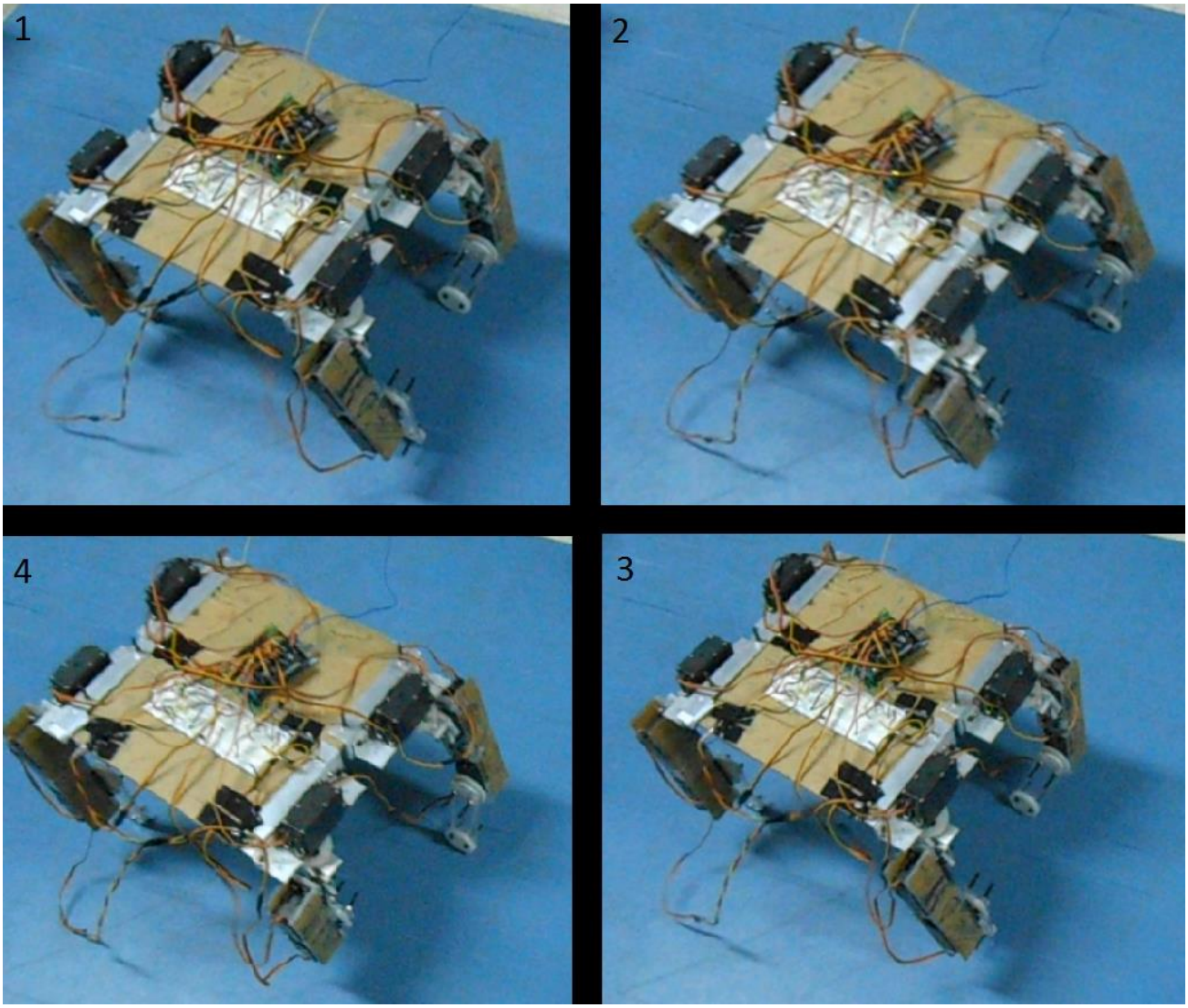


Figure 5.5 Forward-walking gait in quadruped, (frames presented in chronological order, i.e. 1 – 2 – 3 – 4)

As shown in Fig 5.5, the forward walking trot gait has been applied to the quadruped. Just as the crab gait discussed earlier, two legs that are diagonally opposite leave the ground simultaneously whereas the other two provide support. However, what was seen in practice was that stability of the robot in the trot gait is a matter of concern. Therefore, the complete cycle of an individual leg wherein it lifts from the ground and then comes down to act as the stance leg was slightly modified so as to increase the stability. The time taken for it to rise has been decreased to reduce the chances of a fall because then it is only on the support of 2 legs. Fig 5.6 depicts the plot of hip and knee-angles with time. It can be noticed that the Hip yaw angle remains zero throughout the walk cycle. The Arduino code for this robot can be found in Appendix IV.

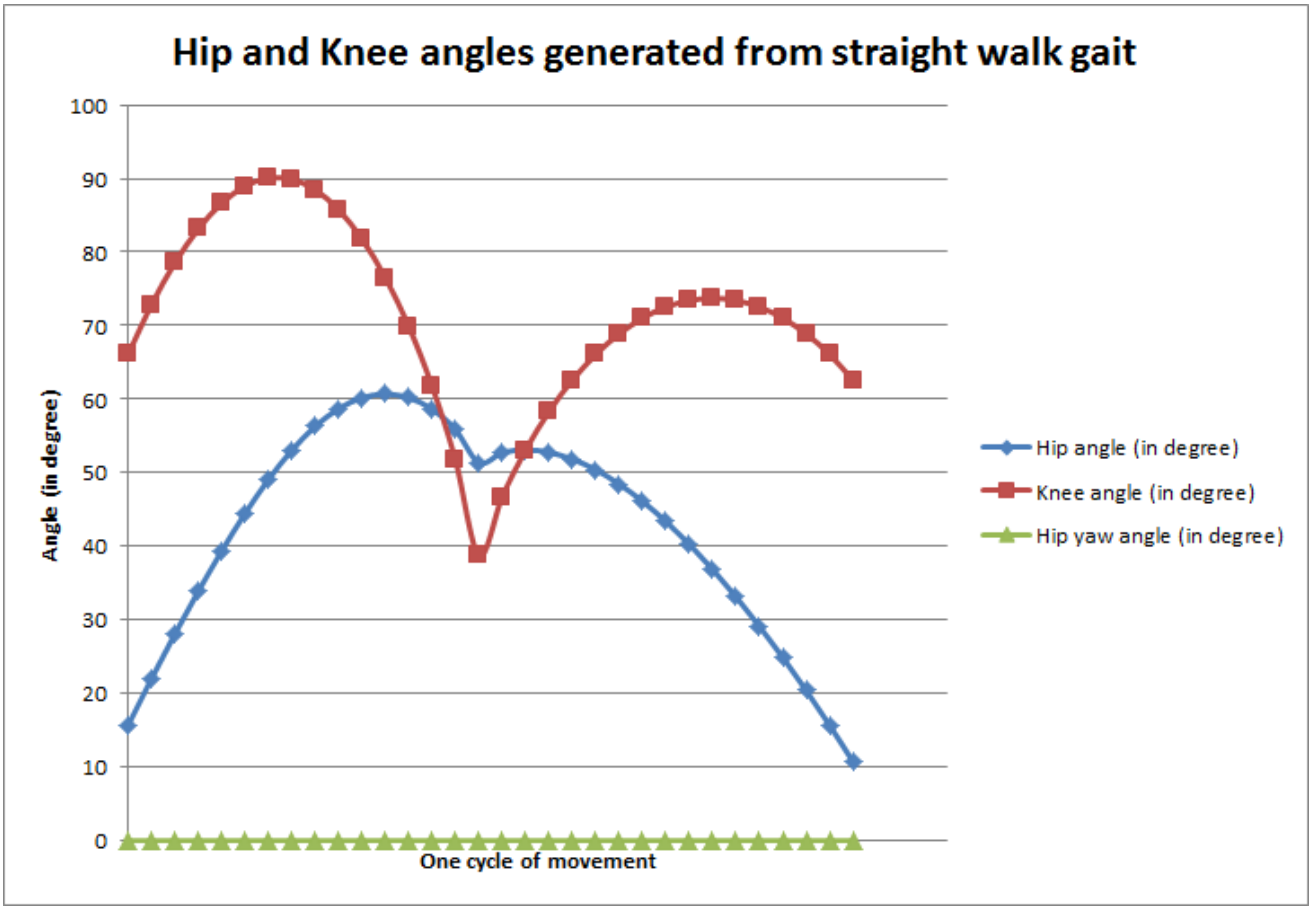


Figure 5.6 Plot of hip and knee-angles with time

5.2.2 Turning trot gait

In order to allow the robot to move around, it is important for it to be able to turn. Therefore, the turning gait needed to be applied and performed on the robot as well. To implement the turning gait, a steering type of method has been used.

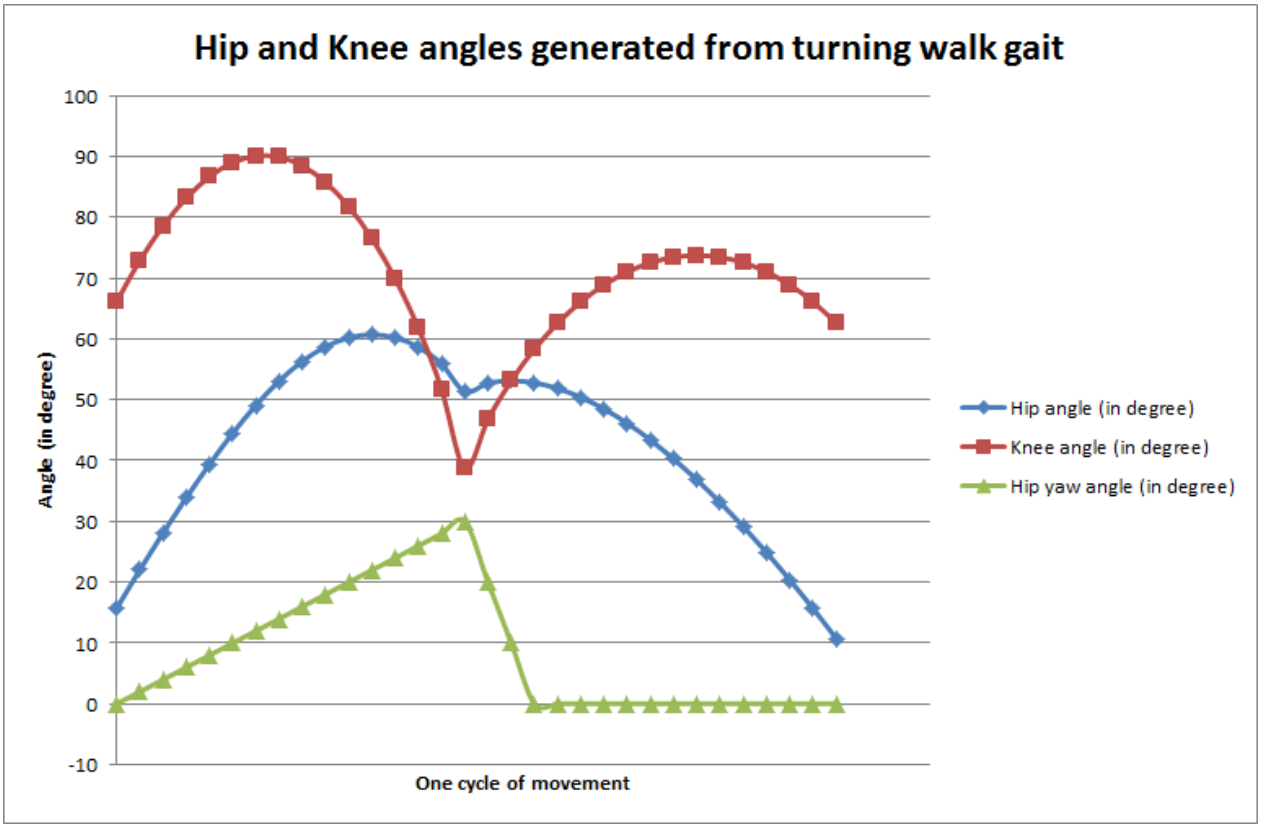


Figure 5.7 Hip, knee and yaw-angles for fore legs from turning gait

In this method the fore legs are primarily responsible for turning. In the fore legs the gait profile is given in Fig 5.7. So for the fore legs the hip yaw angle increases steadily as the leg is in its stride phase and the decreases immediately in its push back phase. However the hind legs follow the normal walking gait as seen in Fig 5.6. Hence by the steering action of the fore legs, the entire robot is able to make a turn while trotting.

This method of turning, in experiments, was found to be much stable than the on the spot turning gaits. Moreover this method allows for variability in radius of turn by altering the yaw angle in a different manner.

Fig 5.8 demonstrates the experiment conducted on the quadruped with regard to the turning trot gait.

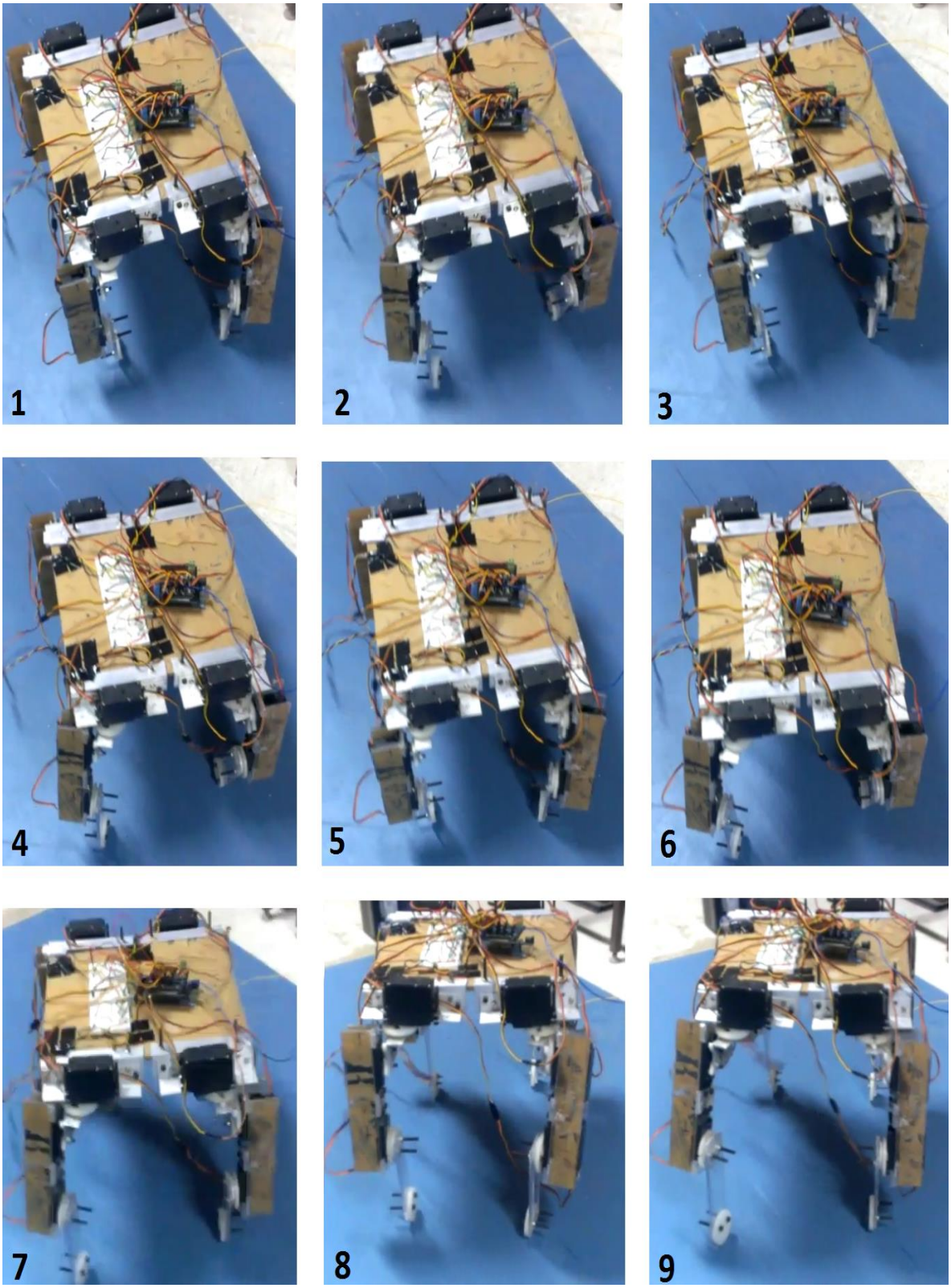


Figure 5.8 Turning-gait on the quadruped; front legs experience a yaw rotation that contributes to turning; frames are presented in a chronological order, i.e. 1-2-3-4-5-6-7-8-9

5.3 Motion Planning

For the purpose of demonstrating the motion planning capacity of the robot, a road following rover was implemented which autonomously traverses a predetermined path on the roads of IIT Guwahati. The hardware present on the robot are:

- 4 X 200 rpm motors (12V 1A) with rotary encoders.
- Inertial Measurement Unit with magnetometer to get accurate heading angle.
- GPS module using the MediaTek MT3329 Chipset gives locational accuracy up-to 10m.
- Ultrasonic transducers for distance sensing from the obstacles. The obstacles are in the form of road pavement which helps in demarcating the road.
- Arduino Uno controller with motor driver circuit.
- 11.1V 2000mAh Lithium Polymer batteries to provide on board power supply.

The main structure of this particular mobile robot or rover has been purchased online. Only the physical framework of the robot was pre-made and has not been manufactured. However, all the requisite sensors, which comprises of the inertial measurement unit (IMU), GPS module, ultrasonic transducers and the microcontroller have been purchased independently, added and assembled to construct this mobile robot.

5.3.1 Quadruped versus Rover for motion planning

There are a multiplicity of reasons as to why motion planning has been executed on a rover as opposed to the quadruped –

- Power requirement:

  For the quadruped to function minus the external power supply, it would need a very powerful battery able to power all the 12 motors simultaneously. Such a high power requirement is monetarily infeasible and also introduces unnecessary weight that would complicate the dynamics of the quadruped to an unfathomable degree.
- Size:

  The quadruped being approximately 35cm high towers above the sidewalks that would be used to sense the left-side margin of the road. Because of its big size and weight associated, it becomes a complicated task to ensure that the quadruped would be able to navigate through all the real-world environments with ease.

Majorly due to the above two reasons, the task of testing the concept of motion planning for quadruped was substituted by the small rover because of its small size, less weight and less power requirements.

5.3.2 Control Flow

The application of motion planning begins with transmittance of the GPS coordinates of the desired path to the rover. Thereafter, motion planning techniques such as potential field algorithms (refer to section 2.7.3) are applied. A potential field algorithm comprises of assigning positive and negative charges to objects within the map. Obstacles are given negative charges whereas the desired position is given a positive one. The robot is attracted to the positive and repelled by the negative which is exactly how potential functions operate. The control flow chart displays the general flow of processes espoused by the rover while navigating, as can be seen in Fig 5.9.

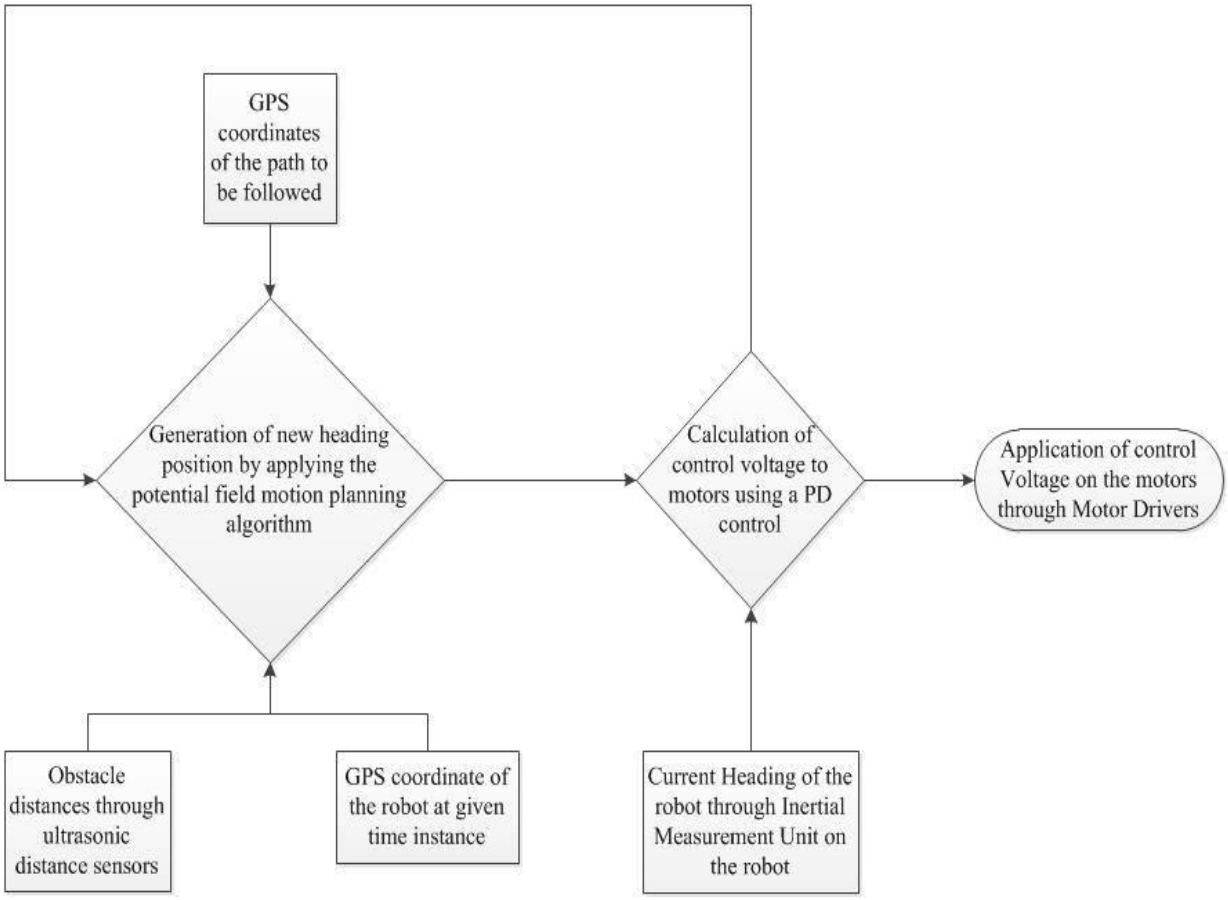


Figure 5.9 Control Flow Chart

In this approach, the information regarding obstacles which are detected using the ultrasonic transducers, as well as the GPS coordinates of the path and the current position, are supplied to the

robot. Thereafter, the potential field function calculates the required new heading position whereas the Inertial Measurement Unit (IMU) senses the current heading position. The difference between the new and current heading position serves as the error for the PD control, which acts upon that error using suitable proportional and derivative gains. The final step of the process results in the application of control torques that are applied to the motors to correct their heading position according to the error. Usually, it is possible in such a function to step into local minima which has been avoided by applying it only in the local environment, thereby reducing the chances of it stepping into a local minima.

Figure 5.10 GPS Map of robot

The path to be followed can be obtained from any open source mapping software. For the purpose of experimentation, the Google maps data has been used. The particular GPS map used can be seen in Fig 5.10.

Here the red path shows the GPS path to be followed (this road path followed was taken from the GPS module mounted on the robot). The vacillating path as depicted is due to the low locational accuracy (10m) of the GPS module. The function of the GPS module is to provide the global location of the robot. Therefore for it to traverse this pre-determined path, it needs sensors like the

Ultrasonic transducers so as to determine obstacles having close proximity with greater accuracy and also help maintain a constant path on that road while keeping a certain distance from the side of the road. The codes associated with the motion planning have been mentioned in the appendix. Fig 5.11 shows the rover traversing the particular path ensuring left lane path following.

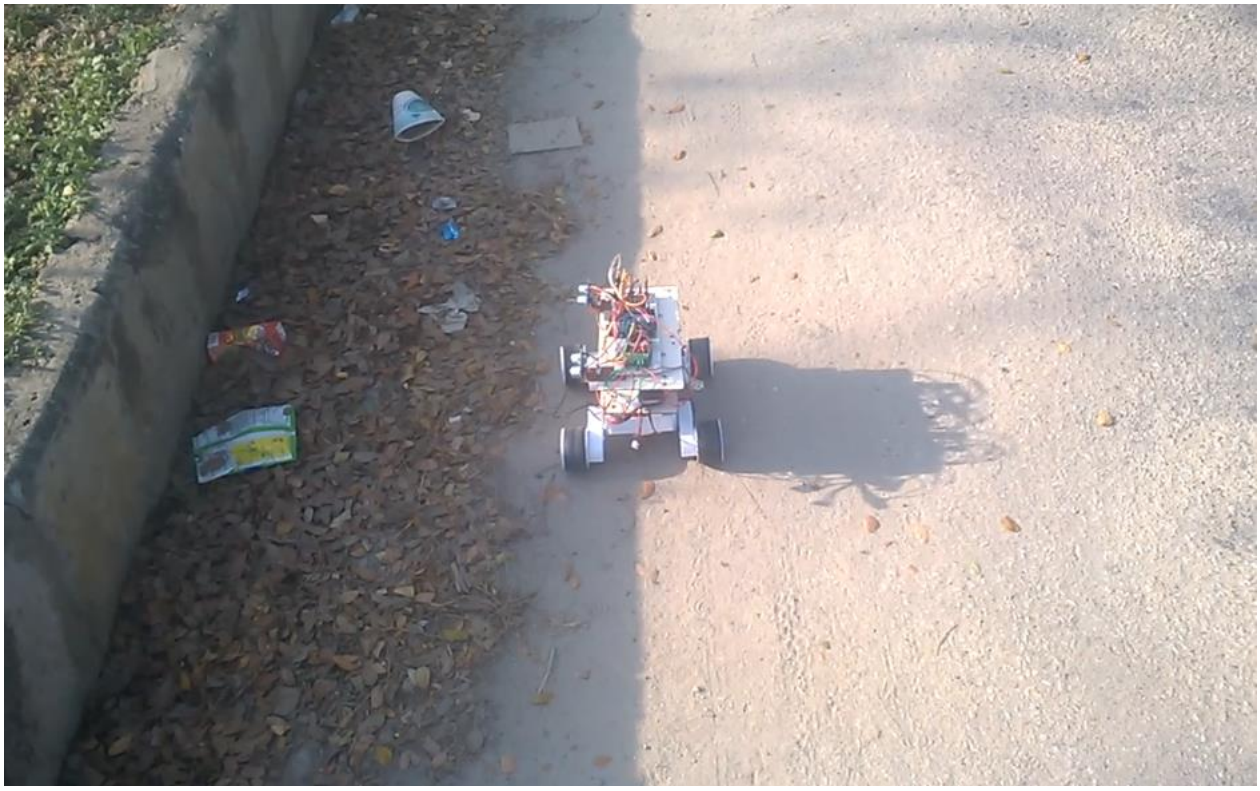


Figure 5.11 Left-lane path following

5.4 Road tracking

As was observed in the previous section, the inability to use the quadruped without the external power supply led to the usage of the rover for motion planning. The requirement of the robot to be self-sufficient in matters of battery, processing units or sensors, necessitates the need to conduct all these operations onboard so as to achieve the objective of attaining self-sufficiency. Which is why it is essential for the robot to be able to function independently without deriving its power or processing from outside the robot.

For a larger scale application, Image processing via an onboard camera is required to demarcate the road or path to be traversed on. Moreover object detection and real world obstacle avoidance would involve several cameras. By implementing Computer Vision techniques of blurring and edge detection, road detection was successfully implemented. Figure 5.12 depicts the road which was to

be filtered and Fig 5.13 shows the results of the road detection code, which provides information of the slope of the road. With this information, the relative position of the robot with respect to the road is calculated. The road detection code can be found in Appendix IV.

Figure 5.12 Sample image for detection

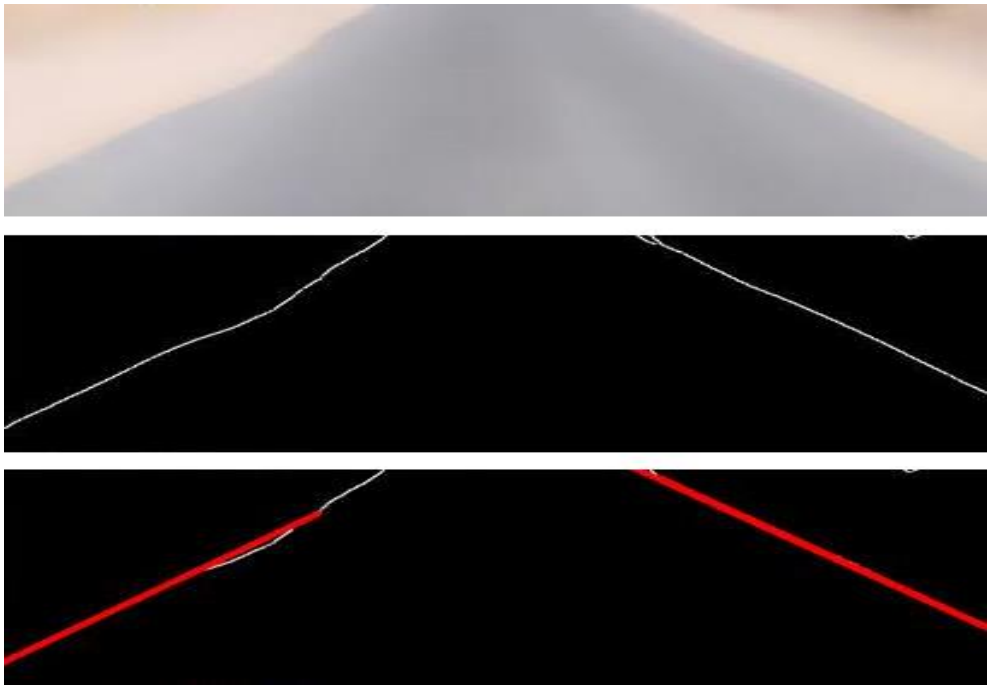


Figure 5.13 Road detection techniques applied on sample road progressively

By using an onboard camera, it is possible to detect the free road surface ahead of the ego-vehicle. Road detection is of high relevance for autonomous driving, road departure warning, and supporting driver-assistance systems such as vehicle and pedestrian detection. The key for vision-

based road detection is the ability to classify image pixels as belonging or not to the road surface. Identifying road pixels is a major challenge due to the intra-class variability caused by lighting conditions. A particularly difficult scenario appears when the road surface has both shadowed and non-shadowed areas. All these aspects have been dealt with, and the hindrances posed by the challenges elaborated previously have been handled to ensure detection of the edge to classify an entity as a road.

Chapter 6

CONCLUSIONS AND DISCUSSIONS

The primary motivation of this research has been to develop a quadruped that would incorporate paradigms pertinent to stability, efficiency and motion planning. Beginning with understanding the concepts crucial to this research, the fundamental aim has been to design and develop a quadruped wherein the same concepts are applied and executed to its full extent.

In this report, we did an extensive literature review to understand the rudimentary concepts related to passivity, stability and motion planning. Concomitantly, a 12-DOF quadruped was designed by first analyzing its preliminary structure, fine-tuning its gait and thereafter, building its CAD model. After the quadruped was assembled, multiple experiments were conducted to establish its gait and improve its stability with respect to different walking patterns. The purpose of this project was to also include motion planning as a part of the experiments on the quadruped, however, due to a need ensure that the robot is economical, those experiments on the quadruped had to be substituted by a rover.

Future improvements for the same would include:

1. Including a Laser Range finder as a part of the robot so as to ensure effective motion planning. The laser distance sensors provide accurate 3D environment data and can allow a much powerful motion planning algorithm
2. Minimize power requirements so as to install a battery onto the system that would help remove dependence on external power supply. Moreover it was found that the Servo motor shafts are not very rigid which reduces the stability of the robot.
3. Deliberate further upon object detection techniques and how such vision-based navigation systems may be incorporated into the quadruped so as to make it more self-sufficient. Stereo vision techniques for effective identification of other vehicles and people on the road will make the system more robust. However to implement such an arrangement on a quadruped, a more powerful processor like the NI Compact RIO should be used.

4 Experimental analysis of passivity can be performed on the created quadruped to obtain energy efficient gaits.

5 Since the control strategies applied in this work are non-adaptive, research into adaptive gait strategies for quadruped could improve efficiency of the robot

REFERENCES

[1] Vukobratovic, Miomir, A. A. Frank, and Davor Juricic. "On the stability of biped locomotion." *Biomedical Engineering, IEEE Transactions on* 1 (1970): 25-36.

[2] Collins, Steve, et al. "Efficient bipedal robots based on passive-dynamic walkers." *Science* 307.5712 (2005): 1082-1085.

[3] http://www.elysium-labs.com/

[4] Vukobratović, Miomir, and Branislav Borovac. "Zero-moment point—thirty five years of its life." *International Journal of Humanoid Robotics* 1.01 (2004): 157-173.

[5] Goswami, Ambarish. "Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 1. IEEE, 1999.

[6] Hurmuzlu, Yildirim, Frank Génot, and Bernard Brogliato. "Modeling, stability and control of biped robots—a general framework." *Automatica* 40.10 (2004): 1647-1664.

[7] Wieber, Pierre-Brice. "On the stability of walking systems." *Proceedings of the international workshop on humanoid and human friendly robotics*. 2002.

[8] Sakagami, Yoshiaki, et al. "The intelligent ASIMO: System overview and integration." *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 3. IEEE, 2002.

[9] Ishida, Toru. "Development of a small biped entertainment robot QRIO." *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004. Proceedings of the 2004 International Symposium on*. IEEE, 2004.

[10] Hirukawa, Hirohisa, et al. "Humanoid robotics platforms developed in HRP." *Robotics and Autonomous Systems* 48.4 (2004): 165-175.

[11] Strogatz, Steven. "Nonlinear dynamics and chaos: with applications to physics, biology, chemistry and engineering." (2001).

[12] McGeer, Tad. "Passive dynamic walking." *the international journal of robotics research* 9.2 (1990): 62-82.

[13] McGeer, Tad. "Passive walking with knees." *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990.

[14] McGeer, Tad. "Stability and control of two-dimensional biped walking." *Center for Systems Science, Simon Fraser University, Burnaby, BC, Canada, Tech. Rep* 1 (1988).

[15] Raibert, Marc H. *Legged robots that balance*. Vol. 3. Cambridge, MA: MIT press, 1986.

[16] Coleman, Michael J., and Andy Ruina. "An uncontrolled walking toy that cannot stand still." *Physical Review Letters* 80.16 (1998): 3658.

[17] Collins, Steven H., Martijn Wisse, and Andy Ruina. "A three-dimensional passive-dynamic walking robot with two legs and knees." *The International Journal of Robotics Research* 20.7 (2001): 607-615.

[18] Mayer, Norbert M., et al. "Stabilizing dynamic walking with physical tricks." *Climbing and Walking Robots*. Springer Berlin Heidelberg, 2005. 835-842.

[19] Tedrake, Russ, et al. "Actuating a simple 3D passive dynamic walker." *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 5. IEEE, 2004.

[20] Ikemata, Yoshito, Akihito Sano, and Hideo Fujimoto. "A physical principle of gait generation and its stabilization derived from mechanism of fixed point." *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006.

[21] Spong, Mark W. "Swing up control of the acrobot." *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994.

[22] Anderson, Stuart O., et al. "Powered bipeds based on passive dynamic principles." *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. IEEE, 2005.

[23] Morimoto, Jun, et al. "A simple reinforcement learning algorithm for biped walking." *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 3. IEEE, 2004.

[24] Endo, Gen, et al. "An empirical exploration of a neural oscillator for biped locomotion control." *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 3. IEEE, 2004.

[25] Chevallereau, Christine, et al. "RABBIT: A testbed for advanced control theory." *IEEE Control Systems Magazine* 23.5 (2003): 57-79.

[26] Westervelt, Eric R., Jessy W. Grizzle, and Daniel E. Koditschek. "Hybrid zero dynamics of planar biped walkers." *Automatic Control, IEEE Transactions on* 48.1 (2003): 42-56.

[27] Canudas-de-Wit, Carlos. "On the concept of virtual constraints as a tool for walking robot control and balancing." *Annual Reviews in Control* 28.2 (2004): 157-166.

[28] Collins, Steven H., and Andy Ruina. "A bipedal walking robot with efficient and human-like gait." *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005.

[29] Wisse, Martijn, and Jan Van Frankenhuyzen. "Design and construction of mike; a 2d autonomous biped based on passive dynamic walking." *Proceedings of International Symposium of Adaptive Motion and Animals and Machines (AMAM03*. 2003.

[30] Wisse, Martijn, Daan GE Hobbelen, and Arend L. Schwab. "Adding an upper body to passive dynamic walking robots by means of a bisecting hip mechanism." *Robotics, IEEE Transactions on* 23.1 (2007): 112-123.

[31] Tedrake, Russ, Teresa Weirui Zhang, and H. Sebastian Seung. "Stochastic policy gradient reinforcement learning on a simple 3D biped." *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE, 2004.

[32] Pratt, Jerry, et al. "Virtual model control: An intuitive approach for bipedal locomotion." *The International Journal of Robotics Research* 20.2 (2001): 129-143.

[33] Pratt, Gill A., and Matthew M. Williamson. "Series elastic actuators." *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*. Vol. 1. IEEE, 1995.

[34] Hutter, M. A. R. C. O., et al. "StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion." *Int. Conf. on Climbing and Walking Robots (CLAWAR)*. 2012.

[35] Raibert, Marc, et al. "Bigdog, the rough-terrain quadruped robot." *Proceedings of the 17th World Congress*. 2008.

[36] Poulakakis, Ioannis, Evangelos Papadopoulos, and Martin Buehler. "On the stable passive dynamics of quadrupedal running." *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 1. IEEE, 2003.

[37] Papadopoulos, Didier, and Martin Buehler. "Stable running in a quadruped robot with compliant legs." *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 1. IEEE, 2000.

[38] Remy, C. David, Keith Buffinton, and Roland Siegwart. "Stability analysis of passive dynamic walking of quadrupeds." *The International Journal of Robotics Research* 29.9 (2010): 1173-1185.

[39] Gan, Zhenyu, and C. David Remy. "A simplistic model for quadrupedal walking and trotting."

[40] Remy, C. David, Keith Buffinton, and Roland Siegwart. "Energetics of passivity–based running with high–compliance series elastic actuation." *International Journal of Mechatronics and Manufacturing Systems* 5.2 (2012): 120-134.

[41] HUTTER, MARCO, et al. "Improved Efficiency in Legged Running Using Lightweight Passive Compliant Feet." (2012).

[42] Hutter, Marco, et al. "Efficient and Versatile Locomotion With Highly Compliant Legs." (2012): 1-10.

[43] Remy, C. David, et al. "Walking and crawling with ALoF: a robot for autonomous locomotion on four legs." *Industrial Robot: An International Journal* 38.3 (2011): 264-268.

[44] Ambe, Yuichi, and Fumitoshi Matsuno. "Leg-grope-walk—Walking strategy on weak and irregular slopes for a quadruped robot by force distribution." *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[45] Siciliano, Bruno, and Oussama Khatib, eds. *Springer handbook of robotics*. Springer, 2008.

[46] LaValle, Steven M. "Motion planning." *Robotics & Automation Magazine, IEEE* 18.2 (2011): 108-118.

[47] Choset, Howie M., ed. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[48] Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *Robotics and Automation, IEEE Transactions on* 12.4 (1996): 566-580.

[49] Amato, Nancy M., O. Burchan Bayazit, and Lucia K. Dale. "OBPRM: An obstacle-based PRM for 3D workspaces." (1998).

[50] Holleman, Christopher, and Lydia E. Kavraki. "A framework for using the workspace medial axis in PRM planners." *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 2. IEEE, 2000.

[51] LaValle, Steven M., Michael S. Branicky, and Stephen R. Lindemann. "On the relationship between classical grid search and probabilistic roadmaps." *The International Journal of Robotics Research* 23.7-8 (2004): 673-692.

[52] Geraerts, Roland, and Mark H. Overmars. "A comparative study of probabilistic roadmap planners." *Algorithmic Foundations of Robotics V*. Springer Berlin Heidelberg, 2004. 43-58.

[53] LaValle, Steven M., and James J. Kuffner Jr. "Rapidly-exploring random trees: Progress and prospects." (2000).

[54] Bekris, Kostas E., et al. "Multiple query probabilistic roadmap planning using single query planning primitives." *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 1. IEEE, 2003.

[55] Cohen, Benjamin J., et al. "Planning for manipulation with adaptive motion primitives." *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011.

[56] Şucan, Ioan A., and Lydia E. Kavraki. "Kinodynamic motion planning by interior-exterior cell exploration." *Algorithmic Foundation of Robotics VIII*. Springer Berlin Heidelberg, 2009. 449-464.

[57] Plaku, Erion, and Gregory D. Hager. "Sampling-based motion and symbolic action planning with geometric and differential constraints." *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010.

[58] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." *The international journal of robotics research* 5.1 (1986): 90-98.

[59] Borenstein, Johann, and Yoram Koren. "Real-time obstacle avoidance for fast mobile robots in cluttered environments." *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990.

[60] LaValle, Steven Michael. *Planning algorithms*. Cambridge university press, 2006.

[61] Van Den Berg, Jur P., and Mark H. Overmars. "Roadmap-based motion planning in dynamic environments." *Robotics, IEEE Transactions on* 21.5 (2005): 885-897.

[62] Karaman, Sertac, et al. "Anytime motion planning using the RRT*." *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011.

[63] Jaillet, Léonard, and Thieny Simeon. "A PRM-based motion planner for dynamically changing environments." *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 2. IEEE, 2004.

[64] Quinlan, Sean, and Oussama Khatib. "Elastic bands: Connecting path planning and control." *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993.

[65] Ko, Nak Yong, Reid G. Simmons, and Dong Jin Seo. "Trajectory modification using elastic force for collision avoidance of a mobile manipulator." *PRICAI 2006: Trends in Artificial Intelligence*. Springer Berlin Heidelberg, 2006. 190-199.

[66] Delsart, Vivien, and Thierry Fraichard. "Navigating dynamic environments using trajectory deformation." *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008.

[67] Gayle, Russell, et al. "Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments." *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007.

[68] Leven, Peter, and Seth Hutchinson. "A framework for real-time path planning in changing environments." *The International Journal of Robotics Research* 21.12 (2002): 999-1030.

[69] Kunz, Tobias, et al. "Real-time path planning for a robot arm in changing environments." *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010.

[70] Chen, Vanessa E. Hsu. *Passive dynamic walking with knees: A point foot model*. Diss. Massachusetts Institute of Technology, 2007.

APPENDIX

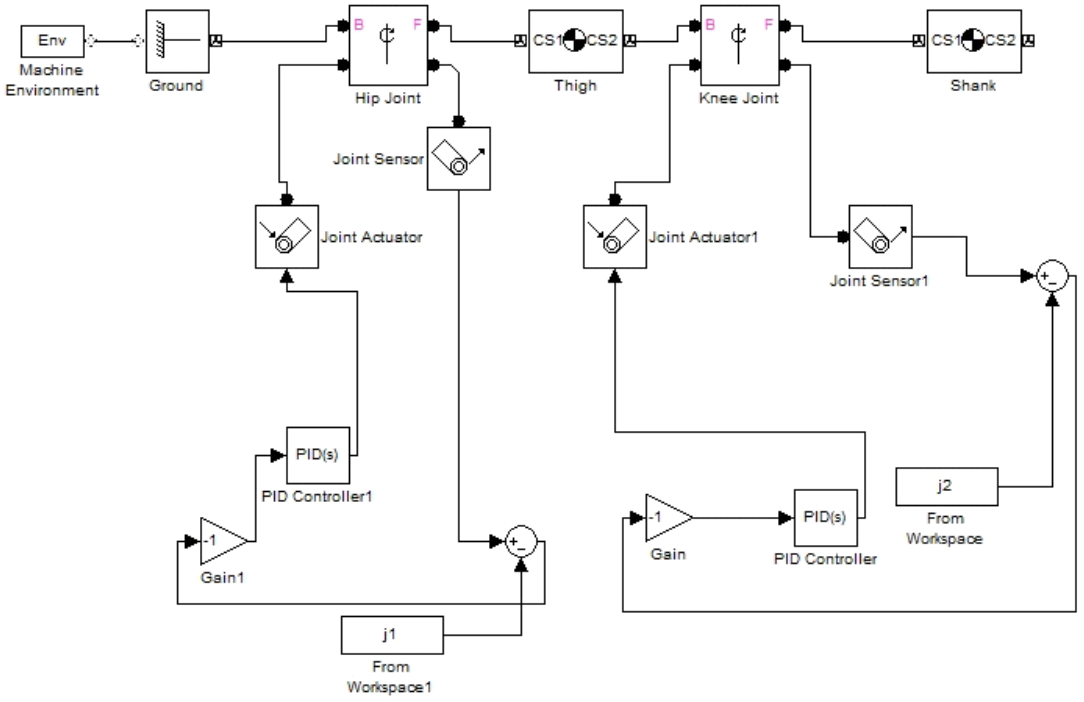
Appendix I.

Control for two linked leg following the gait pattern using Matlab 7.10.0.

a. Data Generation

```
a=0.15;
b=0.15;

R=sqrt(a^2 + b^2)/sqrt(2);

x=R*(-0.5:0.1:1);
y= R*(sin(((x+0.5*R)./(1.5*R))*pi));

x(17:32) = (0.9:-0.1:-0.6)*R;
y(17:32) = 0;
y = y/5;
y = y - 1.6*R;

r=sqrt((x.^2 + y.^2));
phi = atan2(-y,x);
for i = 1:32
   alpha(i) = acos((a.^2 + r(i).^2 - b.^2)/(2*a*r(i)));
   beta(i) = acos((b.^2 + r(i).^2 - a.^2)/(2*b*r(i)));
end

joint1 = phi - alpha;
joint2 = alpha + beta;

joint1 = (joint1')*180/pi;
joint2 = (joint2')*180/pi;

j1 = [0 0];
j1(2,1) = 10;
j1(2,2) = joint1(1);

j2 = [0 0];
j2(2,1) = 10;
j2(2,2) = joint2(1);

j1(3:33,2) = joint1(2:32);
j2(3:33,2) = joint2(2:32);

j1(3:33,1) = (1:31)/2 + 10;
j2(3:33,1) = (1:31)/2 + 10;
```

b. Simulink model


Env
Machine
Environment
Ground
B
F
Hip Joint
CS1
CS2
Thigh
B
F
Knee Joint
CS1
CS2
Shank
Joint Sensor
Joint Actuator
Joint Actuator1
Joint Sensor1
PID(s)
PID Controller1
-1
Gain1
j1
From
Workspace1
-1
Gain
PID(s)
PID Controller
j2
From
Workspace

Appendix II.

To obtain quadruped gait simulation on Matlab 7.10.0

a.) data generation code

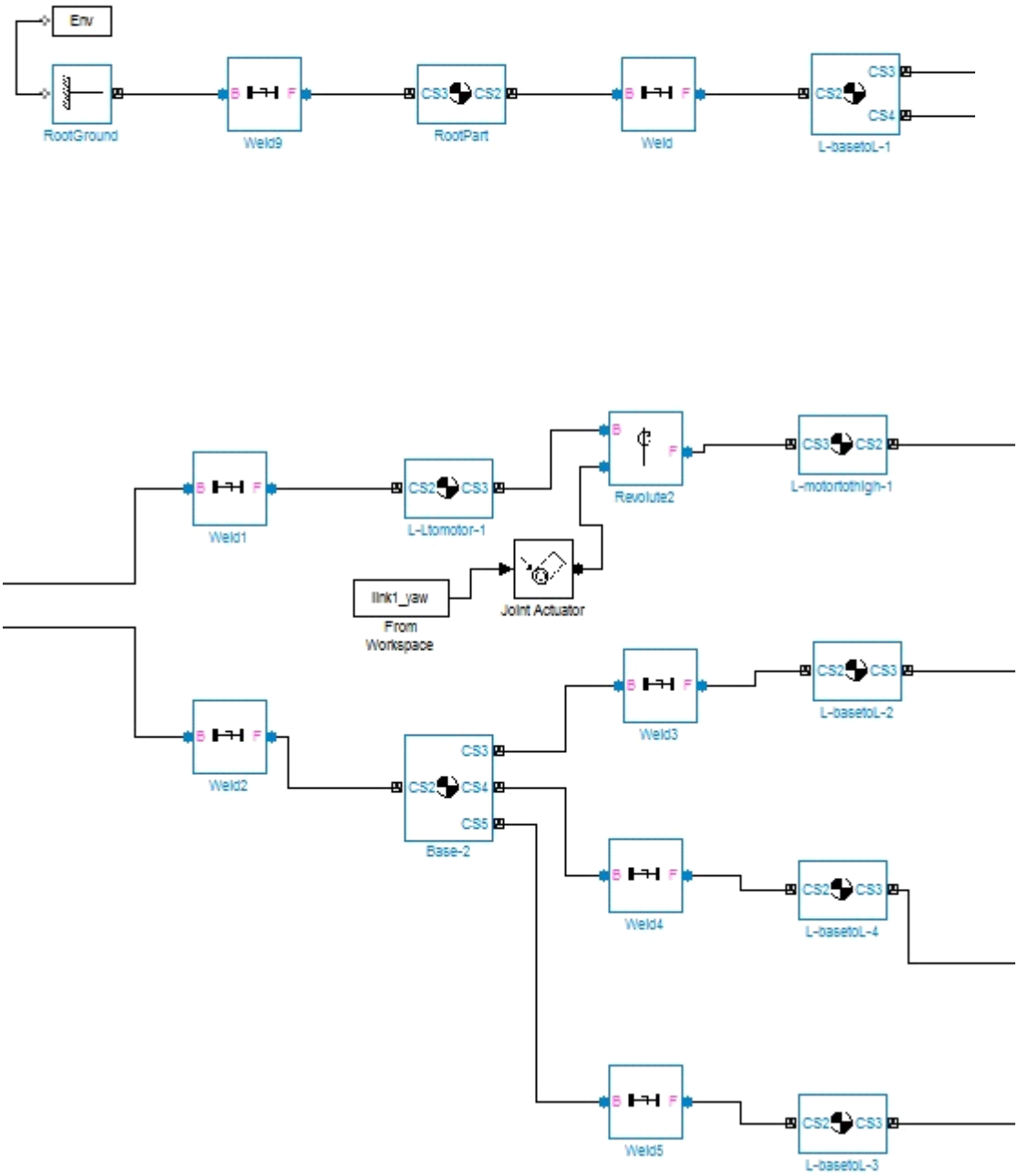
```
clear;

% joint_data_simulation.mat contains variable j1 and j2 which contain
% gait data. j1 contains the hip angle data vector with equally spaced (temporally)
% data while j2 contains the knee angle data vector with equally spaced data.

joints=open('joint_data_simulation.mat');

simulation_time = 1000;

forward_hip = joints.j1(1:16);
backward_hip = joints.j1(17:32);
forward_knee = joints.j2(1:16);
backward_knee = joints.j2(17:32);

forward_start_hip = ones(16,1)*joints.j1(1);
backward_start_hip = ones(16,1)*joints.j1(16);
forward_start_knee = ones(16,1)*joints.j2(1);
backward_start_knee = ones(16,1)*joints.j2(16);

link1_yaw.time=0:1:simulation_time-1;
link1_yaw.signals.dimensions=3;
link1_yaw.signals.values=0*ones(simulation_time,3);

link1_hip.time=0:1:simulation_time-1;
link1_hip.signals.dimensions=3;
link1_hip.signals.values=-0*ones(simulation_time,3);

link1_knee.time=0:1:simulation_time-1;
link1_knee.signals.dimensions=3;
link1_knee.signals.values=0*ones(simulation_time,3);




link2_yaw.time=0:1:simulation_time-1;
link2_yaw.signals.dimensions=3;
link2_yaw.signals.values=0*ones(simulation_time,3);

link2_hip.time=0:1:simulation_time-1;
link2_hip.signals.dimensions=3;
link2_hip.signals.values=0*ones(simulation_time,3);

link2_knee.time=0:1:simulation_time-1;
link2_knee.signals.dimensions=3;
link2_knee.signals.values=0*ones(simulation_time,3);
```

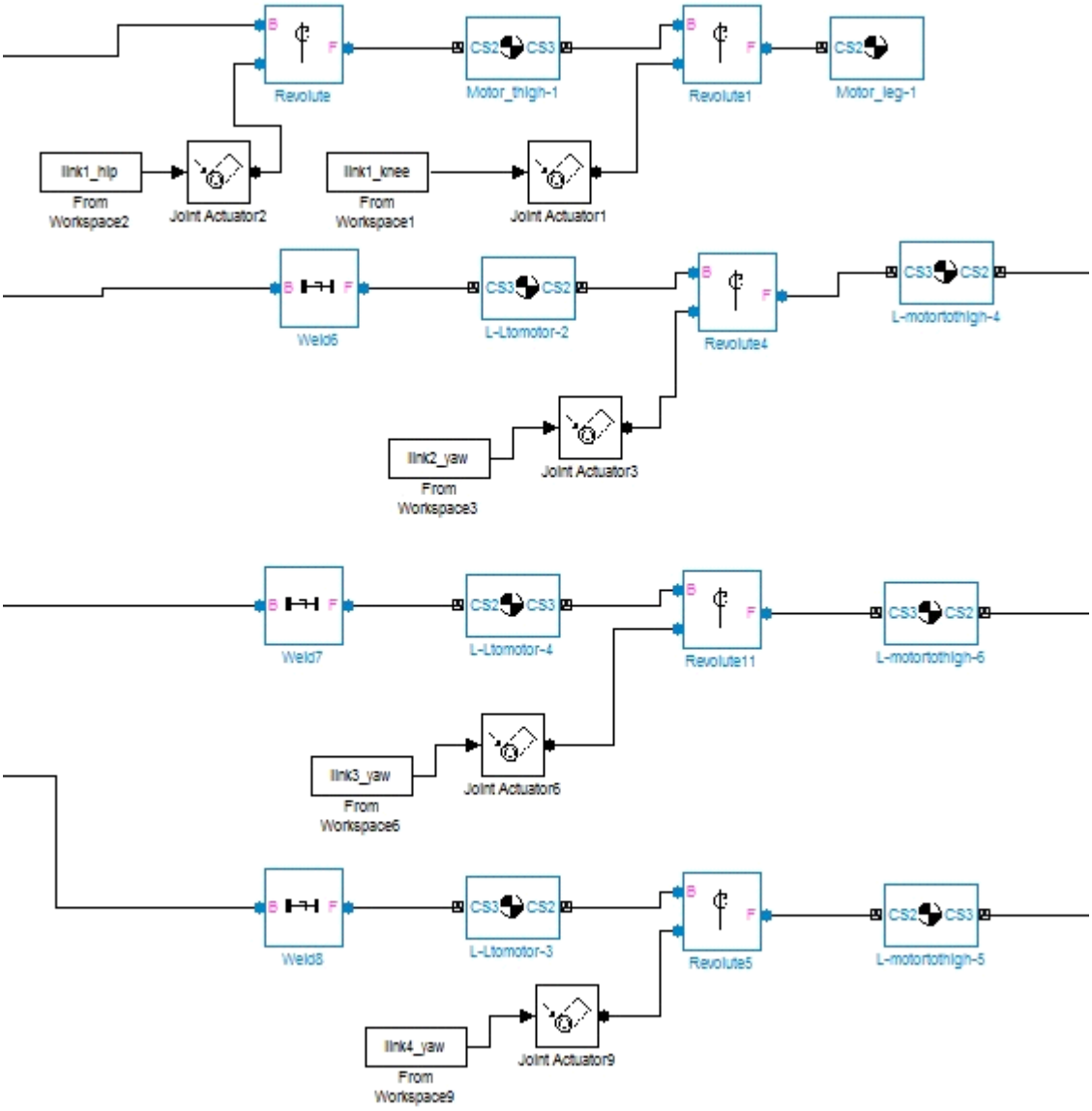
```
link3_yaw.time=0:1:simulation_time-1;
link3_yaw.signals.dimensions=3;
link3_yaw.signals.values=0*ones(simulation_time,3);

link3_hip.time=0:1:simulation_time-1;
link3_hip.signals.dimensions=3;
link3_hip.signals.values=-0*ones(simulation_time,3);

link3_knee.time=0:1:simulation_time-1;
link3_knee.signals.dimensions=3;
link3_knee.signals.values=0*ones(simulation_time,3);




link4_yaw.time=0:1:simulation_time-1;
link4_yaw.signals.dimensions=3;
link4_yaw.signals.values=0*ones(simulation_time,3);

link4_hip.time=0:1:simulation_time-1;
link4_hip.signals.dimensions=3;
link4_hip.signals.values=0*ones(simulation_time,3);

link4_knee.time=0:1:simulation_time-1;
link4_knee.signals.dimensions=3;
link4_knee.signals.values=0*ones(simulation_time,3);


j=1;
del = 2;
for i =1:10

   if (i == 1)

      link1_hip.signals.values(j:j+15,1) = -forward_hip;
      link1_knee.signals.values(j:j+15,1) = forward_knee;

      link2_hip.signals.values(j:j+15,1) =  forward_start_hip;
      link2_knee.signals.values(j:j+15,1) = forward_start_knee;

      link3_hip.signals.values(j:j+15,1) = -forward_start_hip;
      link3_knee.signals.values(j:j+15,1) = forward_start_knee;

      link4_hip.signals.values(j:j+15,1) =  forward_hip;
      link4_knee.signals.values(j:j+15,1) = forward_knee;

      j=j+15+1;

      link1_hip.signals.values(j:j+del,1) = -backward_start_hip(1);
      link1_knee.signals.values(j:j+del,1) = backward_start_knee(1);
```

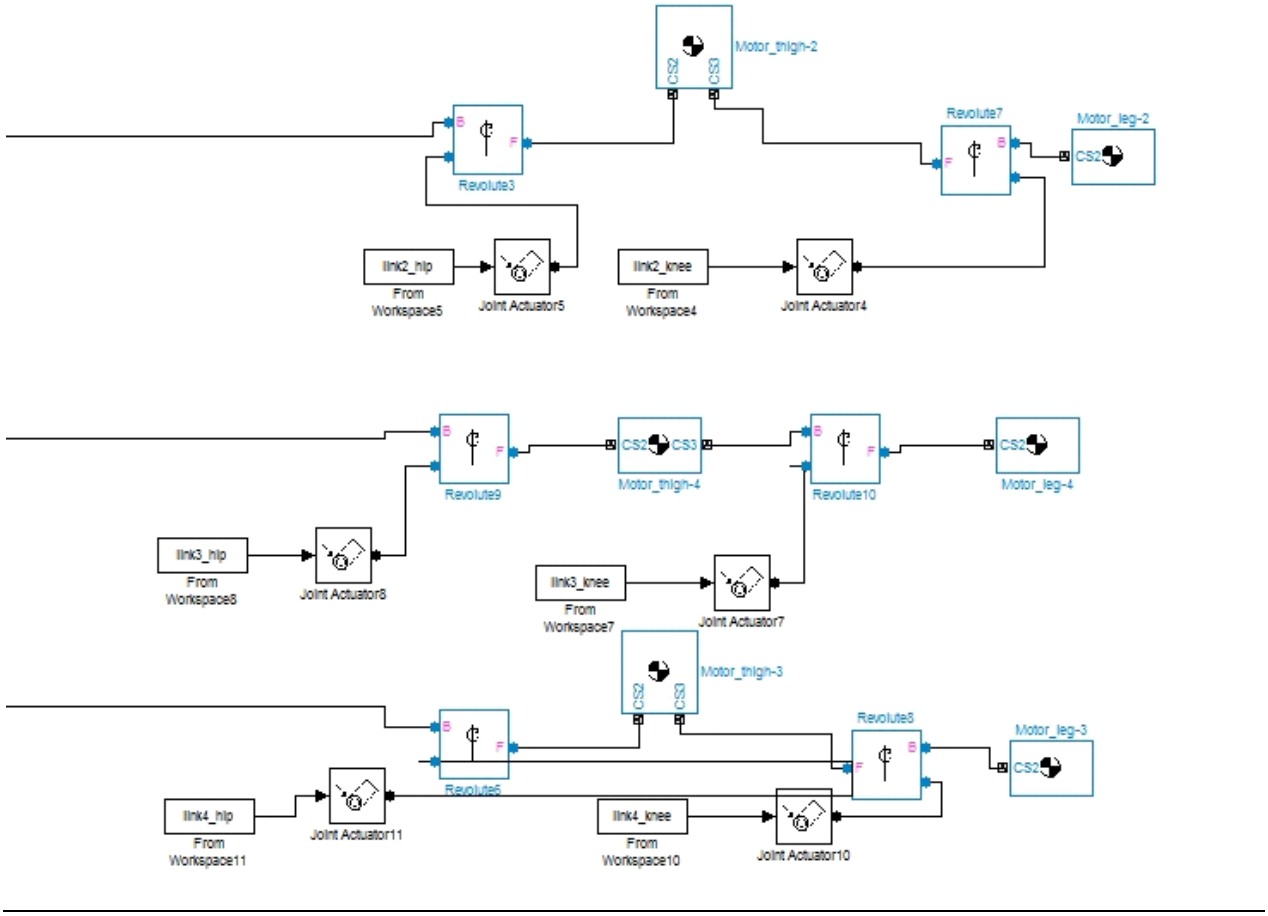
```
    link2_hip.signals.values(j:j+del,1) =  forward_start_hip(1);
    link2_knee.signals.values(j:j+del,1) = forward_start_knee(1);

    link3_hip.signals.values(j:j+del,1) = -forward_start_hip(1);
    link3_knee.signals.values(j:j+del,1) = forward_start_knee(1);

    link4_hip.signals.values(j:j+del,1) =  backward_start_hip(1);
    link4_knee.signals.values(j:j+del,1) = backward_start_knee(1);

    j=j+del+1;
else
    link1_hip.signals.values(j:j+15,1) = -forward_hip;
    link1_knee.signals.values(j:j+15,1) = forward_knee;

    link2_hip.signals.values(j:j+15,1) =  backward_hip;
    link2_knee.signals.values(j:j+15,1) = backward_knee;

    link3_hip.signals.values(j:j+15,1) = -backward_hip;
    link3_knee.signals.values(j:j+15,1) = backward_knee;

    link4_hip.signals.values(j:j+15,1) =  forward_hip;
    link4_knee.signals.values(j:j+15,1) = forward_knee;

    j=j+15+1;

    link1_hip.signals.values(j:j+del,1) = -backward_start_hip(1);
    link1_knee.signals.values(j:j+del,1) = backward_start_knee(1);

    link2_hip.signals.values(j:j+del,1) =  forward_start_hip(1);
    link2_knee.signals.values(j:j+del,1) = forward_start_knee(1);

    link3_hip.signals.values(j:j+del,1) = -forward_start_hip(1);
    link3_knee.signals.values(j:j+del,1) = forward_start_knee(1);

    link4_hip.signals.values(j:j+del,1) =  backward_start_hip(1);
    link4_knee.signals.values(j:j+del,1) = backward_start_knee(1);

    j=j+del+1;
end

    link1_hip.signals.values(j:j+15,1) = -backward_hip;
    link1_knee.signals.values(j:j+15,1) = backward_knee;

    link2_hip.signals.values(j:j+15,1) =  forward_hip;
    link2_knee.signals.values(j:j+15,1) = forward_knee;

    link3_hip.signals.values(j:j+15,1) = -forward_hip;
    link3_knee.signals.values(j:j+15,1) = forward_knee;

    link4_hip.signals.values(j:j+15,1) =  backward_hip;
    link4_knee.signals.values(j:j+15,1) = backward_knee;

    j=j+15+1;

    link1_hip.signals.values(j:j+del,1) = -forward_start_hip(1);
```

```
        link1_knee.signals.values(j:j+del,1) = forward_start_knee(1);

        link2_hip.signals.values(j:j+del,1) =  backward_start_hip(1);
        link2_knee.signals.values(j:j+5,1) = backward_start_knee(1);

        link3_hip.signals.values(j:j+del,1) = -backward_start_hip(1);
        link3_knee.signals.values(j:j+del,1) = backward_start_knee(1);

        link4_hip.signals.values(j:j+del,1) =  forward_start_hip(1);
        link4_knee.signals.values(j:j+del,1) = forward_start_knee(1);

        j=j+del+1;

end
```

b.) Simulink code generated by importing Solidworks model and applying joint actuators on Simulink SimMechanics




Revolute
Motor_thigh-1
Revolute1
Motor_leg-1
link1_hip
From Workspace2
Joint Actuator2
link1_knee
From Workspace1
Joint Actuator1
Weld6
L-Ltomotor-2
Revolute4
L-motortothigh-4
link2_yaw
From Workspace3
Joint Actuator3
Weld7
L-Ltomotor-4
Revolute11
L-motortothigh-6
link3_yaw
From Workspace6
Joint Actuator6
Weld8
L-Ltomotor-3
Revolute5
L-motortothigh-5
link4_yaw
From Workspace9
Joint Actuator9


Motor_thigh-2
CS2
CS3
Revolute3
Revolute7
Motor_leg-2
CS2
llink2_hip
From
Workspace5
Joint Actuator5
llink2_knee
From
Workspace4
Joint Actuator4
Revolute9
CS2
CS3
Motor_thigh-4
Revolute10
CS2
Motor_leg-4
llink3_hip
From
Workspace8
Joint Actuator8
llink3_knee
From
Workspace7
Joint Actuator7
Motor_thigh-3
CS2
CS3
Revolute6
Revolute8
Motor_leg-3
CS2
llink4_hip
From
Workspace11
Joint Actuator11
llink4_knee
From
Workspace10
Joint Actuator10

Appendix III.

The larger quadruped's straight walking and turning code written on the Arduino IDE

```
#include<Servo.h>
#define T 9
#define timeP 500

int timespent = 0;
int del = 10;


//      LEG 1 functions and variables


Servo leg1servo[3];

int leg1zeroservo[]={100,90,90};
int leg1targetservo[]={90, 90, 90};

int leg1angularpath[3][T],leg1targetservopath[3][T];

int leg1angle0[]={0,10,20,20,0,0,0,0,0};
int leg1angle1[]={5,25,25,18,17,15,12,9,5};
int leg1angle2[]={22,22,22,18,22,25,25,25,22};


void leg1createpath()
{
  for( int i=0;i<T;i++)
  {
    leg1angularpath[0][i] = leg1angle0[i];
    leg1angularpath[1][i] = leg1angle1[i];
    leg1angularpath[2][i] = leg1angle2[i];
  }
}

void leg1createtargetservopath()
{
  for( int i=0;i<T;i++)
  {
    leg1targetservopath[0][i] = leg1zeroservo[0] + leg1angle0[i];
    leg1targetservopath[1][i] = leg1zeroservo[1] + leg1angle1[i];
    leg1targetservopath[2][i] = leg1zeroservo[2] + leg1angle2[i];
  }
}

int leg1iszero()
```

```
{

  int i=0, flag =0;
  while(i<3)
  {
    if( leg1servo[i].read() > leg1zeroservo[i]  || leg1servo[i].read() < leg1zeroservo[i]  )
    {
      flag =1;
      break;
    }
    i++;
  }
  return( !flag);
}

int leg1istarget()
{
  int i=0, flag =0;
  while(i<3)
  {
    if( leg1servo[i].read() > leg1targetservo[i] + 1 || leg1servo[i].read() < leg1targetservo[i] - 1 )
    {
      flag =1;
      break;
    }
    i++;
  }

  return( !flag);
}

void leg1gettozero()
{
  timespent = 0;

  while(1)
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg1zeroservo[i]>leg1servo[i].read() )
        leg1servo[i].write(leg1servo[i].read() + 1);

      if( leg1zeroservo[i]<leg1servo[i].read() )
        leg1servo[i].write(leg1servo[i].read() - 1);
    }
    delay(del);
    timespent = timespent + del;
    if ( leg1iszero() )
    {
```

```
     break;
    }
  }

}

void leg1gettotarget()
{
  while(!leg1istarget())
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg1targetservo[i]>leg1servo[i].read() )
        leg1servo[i].write(leg1servo[i].read() + 1);

      if( leg1targetservo[i]<leg1servo[i].read() )
        leg1servo[i].write(leg1servo[i].read() - 1);
    }
    delay(del);
  }
}



//      LEG 2 functions and variables


Servo leg2servo[3];

int leg2zeroservo[]={90,137,95};
int leg2targetservo[]={90, 90, 90};

int leg2angularpath[3][T],leg2targetservopath[3][T];

int leg2angle0[]={0,-10,-20,-20,0,0,0,0,0};
int leg2angle1[]={5,25,25,18,17,15,12,9,5};
int leg2angle2[]={22,22,22,18,22,25,25,25,22};


void leg2createpath()
{
  for( int i=0;i<T;i++)
  {
    leg2angularpath[0][i] = leg2angle0[i];
    leg2angularpath[1][i] = leg2angle1[i];
    leg2angularpath[2][i] = leg2angle2[i];
  }
}
```

```
void leg2createtargetservopath()
{
  for( int i=0;i<T;i++)
  {
    leg2targetservopath[0][i] = leg2zeroservo[0] - leg2angle0[i];
    leg2targetservopath[1][i] = leg2zeroservo[1] - leg2angle1[i];
    leg2targetservopath[2][i] = leg2zeroservo[2] - leg2angle2[i];
  }
}

int leg2iszero()
{

  int i=0, flag =0;
  while(i<3)
  {
    if( leg2servo[i].read() > leg2zeroservo[i]  || leg2servo[i].read() < leg2zeroservo[i]  )
    {
      flag =1;
      break;
    }
    i++;
  }
  return( !flag);
}

int leg2istarget()
{
  int i=0, flag =0;
  while(i<3)
  {
    if( leg2servo[i].read() > leg2targetservo[i] + 1 || leg2servo[i].read() < leg2targetservo[i] - 1 )
    {
      flag =1;
      break;
    }
    i++;
  }

  return( !flag);
}

void leg2gettozero()
{
  timespent = 0;

  while(1)
  {
    for(int i = 0; i < 3;i++)
```

```
    {
      if( leg2zeroservo[i]>leg2servo[i].read() )
        leg2servo[i].write(leg2servo[i].read() + 1);

      if( leg2zeroservo[i]<leg2servo[i].read() )
        leg2servo[i].write(leg2servo[i].read() - 1);
    }
    delay(del);
    timespent = timespent + del;
    if ( leg2iszero() )
    {
      break;
    }
  }

}

void leg2gettotarget()
{
  while(!leg2istarget())
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg2targetservo[i]>leg2servo[i].read() )
        leg2servo[i].write(leg2servo[i].read() + 1);

      if( leg2targetservo[i]<leg2servo[i].read() )
        leg2servo[i].write(leg2servo[i].read() - 1);
    }
    delay(del);
  }
}



//      LEG 2 functions and variables


Servo leg3servo[3];

int leg3zeroservo[]={81,82,85};
int leg3targetservo[]={90, 90, 90};

int leg3angularpath[3][T],leg3targetservopath[3][T];

int leg3angle0[]={0,-10,-20,-20,-16,-12,-8,-4,0};
int leg3angle1[]={5,25,25,18,17,15,12,9,5};
int leg3angle2[]={22,22,22,18,22,25,25,25,22};
```

```
void leg3createpath()
{
  for( int i=0;i<T;i++)
  {
    leg3angle0[i] = 0;
    leg3angularpath[0][i] = leg3angle0[i];
    leg3angularpath[1][i] = leg3angle1[i];
    leg3angularpath[2][i] = leg3angle2[i];
  }
}

void leg3createtargetservopath()
{
  for( int i=0;i<T;i++)
  {
    leg3targetservopath[0][i] = leg3zeroservo[0] - leg3angle0[i];
    leg3targetservopath[1][i] = leg3zeroservo[1] - leg3angle1[i];
    leg3targetservopath[2][i] = leg3zeroservo[2] - leg3angle2[i];
  }
}

int leg3iszero()
{

  int i=0, flag =0;
  while(i<3)
  {
    if( leg3servo[i].read() > leg3zeroservo[i]  || leg3servo[i].read() < leg3zeroservo[i]  )
    {
      flag =1;
      break;
    }
    i++;
  }
  return( !flag);
}

int leg3istarget()
{
  int i=0, flag =0;
  while(i<3)
  {
    if( leg3servo[i].read() > leg3targetservo[i] + 1 || leg3servo[i].read() < leg3targetservo[i] - 1 )
    {
      flag =1;
      break;
    }
    i++;
```

```
  }

  return( !flag);
}

void leg3gettozero()
{
  timespent = 0;

  while(1)
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg3zeroservo[i]>leg3servo[i].read() )
        leg3servo[i].write(leg3servo[i].read() + 1);

      if( leg3zeroservo[i]<leg3servo[i].read() )
        leg3servo[i].write(leg3servo[i].read() - 1);
    }
    delay(del);
    timespent = timespent + del;
    if ( leg3iszero() )
    {
      break;
    }
  }

}

void leg3gettotarget()
{
  while(!leg3istarget())
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg3targetservo[i]>leg3servo[i].read() )
        leg3servo[i].write(leg3servo[i].read() + 1);

      if( leg3targetservo[i]<leg3servo[i].read() )
        leg3servo[i].write(leg3servo[i].read() - 1);
    }
    delay(del);
  }
}
```

```
//      LEG 4 functions and variables


Servo leg4servo[3];

int leg4zeroservo[]={86,90,107};
int leg4targetservo[]={90, 90, 90};

int leg4angularpath[3][T],leg4targetservopath[3][T];

int leg4angle0[]={0,10,20,20,16,12,8,4,0};
int leg4angle1[]={5,25,25,18,17,15,12,9,5};
int leg4angle2[]={22,22,22,18,22,25,25,25,22};


void leg4createpath()
{
  for( int i=0;i<T;i++)
  {
    leg4angle0[i] = 0;
    leg4angularpath[0][i] = leg4angle0[i];
    leg4angularpath[1][i] = leg4angle1[i];
    leg4angularpath[2][i] = leg4angle2[i];
  }
}

void leg4createtargetservopath()
{
  for( int i=0;i<T;i++)
  {
    leg4targetservopath[0][i] = leg4zeroservo[0] + leg4angle0[i];
    leg4targetservopath[1][i] = leg4zeroservo[1] + leg4angle1[i];
    leg4targetservopath[2][i] = leg4zeroservo[2] + leg4angle2[i];
  }
}

int leg4iszero()
{

  int i=0, flag =0;
  while(i<3)
  {
    if( leg4servo[i].read() > leg4zeroservo[i]  || leg4servo[i].read() < leg4zeroservo[i]  )
    {
      flag =1;
      break;
    }
    i++;
  }
```

```
  return( !flag);
}

int leg4istarget()
{
  int i=0, flag =0;
  while(i<3)
  {
    if( leg4servo[i].read() > leg4targetservo[i] + 1 || leg4servo[i].read() < leg4targetservo[i] - 1 )
    {
      flag =1;
      break;
    }
    i++;
  }

  return( !flag);
}

void leg4gettozero()
{
  timespent = 0;

  while(1)
  {
    for(int i = 0; i < 3;i++)
    {
      if( leg4zeroservo[i]>leg4servo[i].read() )
        leg4servo[i].write(leg4servo[i].read() + 1);

      if( leg4zeroservo[i]<leg4servo[i].read() )
        leg4servo[i].write(leg4servo[i].read() - 1);
    }
    delay(del);
    timespent = timespent + del;
    if ( leg4iszero() )
    {
      break;
    }
  }

}

void leg4gettotarget()
{
  while(!leg4istarget())
  {
    for(int i = 0; i < 3;i++)
    {
```

```
     if( leg4targetservo[i]>leg4servo[i].read() )
       leg4servo[i].write(leg4servo[i].read() + 1);

     if( leg4targetservo[i]<leg4servo[i].read() )
       leg4servo[i].write(leg4servo[i].read() - 1);
    }
    delay(del);
  }
}



void setup()
{

  // attachment of LEG1

  leg1servo[0].attach(4);
  leg1servo[1].attach(2);
  leg1servo[2].attach(22);
  Serial.begin(9600);

  leg1createpath();
  leg1createtargetservopath();


  // attachment of LEG2

  leg2servo[0].attach(5);
  leg2servo[1].attach(3);
  leg2servo[2].attach(24);

  leg2createpath();
  leg2createtargetservopath();


  // attachment of leg3

  leg3servo[0].attach(13);
  leg3servo[1].attach(10);
  leg3servo[2].attach(26);

  leg3createpath();
  leg3createtargetservopath();


  // attachment of leg4

  leg4servo[0].attach(6);
```

```
  leg4servo[1].attach(11);
  leg4servo[2].attach(28);

  leg4createpath();
  leg4createtargetservopath();
}

void loop()
{

  /*myservo[0].write(100);
  myservo[1].write(90);
  myservo[2].write(90);
  delay(15);*/

delay(2000);
leg1gettozero();
leg2gettozero();
leg3gettozero();
leg4gettozero();

delay(2000);

leg1targetservo[0] = leg1targetservopath[0][0];
leg1targetservo[1] = leg1targetservopath[1][0];
leg1targetservo[2] = leg1targetservopath[2][0];

leg2targetservo[0] = leg2targetservopath[0][0];
leg2targetservo[1] = leg2targetservopath[1][0];
leg2targetservo[2] = leg2targetservopath[2][0];

leg3targetservo[0] = leg3targetservopath[0][0];
leg3targetservo[1] = leg3targetservopath[1][0];
leg3targetservo[2] = leg3targetservopath[2][0];

leg4targetservo[0] = leg4targetservopath[0][0];
leg4targetservo[1] = leg4targetservopath[1][0];
leg4targetservo[2] = leg4targetservopath[2][0];

leg1gettotarget();
leg2gettotarget();
leg3gettotarget();
leg4gettotarget();

delay(10000);

/*
leg3targetservo[0] = leg3targetservopath[0][0];
     leg3targetservo[1] = leg3targetservopath[1][0] - 10;
```

```
      leg3targetservo[2] = leg3targetservopath[2][0] - 20;

      leg3gettotarget();

      delay(5000);*/
del = 3;
for( int i=0;i<=3;i++)
    {


      leg2targetservo[0] = leg2targetservopath[0][i];
      leg2targetservo[1] = leg2targetservopath[1][i];
      leg2targetservo[2] = leg2targetservopath[2][i];

      leg4targetservo[0] = leg4targetservopath[0][i];
      leg4targetservo[1] = leg4targetservopath[1][i];
      leg4targetservo[2] = leg4targetservopath[2][i];


      leg2gettotarget();

  //    Serial.println(i);
      leg4gettotarget();

      //delay(del);
      //delay(timeP - timespent);
    }

   // while(1);

   delay(1000);

while(1)
{

  for( int i=4;i<T;i++)
    {


      leg2targetservo[0] = leg2targetservopath[0][i];
      leg2targetservo[1] = leg2targetservopath[1][i];
      leg2targetservo[2] = leg2targetservopath[2][i];

      leg4targetservo[0] = leg4targetservopath[0][i];
      leg4targetservo[1] = leg4targetservopath[1][i];
      leg4targetservo[2] = leg4targetservopath[2][i];

      if ( (i-4)<=3 )
      {
```

```
      leg1targetservo[0] = leg1targetservopath[0][i-4];
      leg1targetservo[1] = leg1targetservopath[1][i-4];
      leg1targetservo[2] = leg1targetservopath[2][i-4];

      leg3targetservo[0] = leg3targetservopath[0][i-4];
      leg3targetservo[1] = leg3targetservopath[1][i-4];
      leg3targetservo[2] = leg3targetservopath[2][i-4];
    }

    leg2gettotarget();

    leg4gettotarget();

    leg1gettotarget();

    leg3gettotarget();

    delay(del);
    //delay(timeP - timespent);
  }

delay(1000);

for( int i=4;i<T;i++)
  {


    leg1targetservo[0] = leg1targetservopath[0][i];
    leg1targetservo[1] = leg1targetservopath[1][i];
    leg1targetservo[2] = leg1targetservopath[2][i];

    leg3targetservo[0] = leg3targetservopath[0][i];
    leg3targetservo[1] = leg3targetservopath[1][i];
    leg3targetservo[2] = leg3targetservopath[2][i];

    if ( (i-4)<=3 )
    {
      leg2targetservo[0] = leg2targetservopath[0][i-4];
      leg2targetservo[1] = leg2targetservopath[1][i-4];
      leg2targetservo[2] = leg2targetservopath[2][i-4];

      leg4targetservo[0] = leg4targetservopath[0][i-4];
      leg4targetservo[1] = leg4targetservopath[1][i-4];
      leg4targetservo[2] = leg4targetservopath[2][i-4];
    }

    leg1gettotarget();

    leg3gettotarget();
```

```
      leg2gettotarget();

      leg4gettotarget();

      delay(del);
      //delay(timeP - timespent);
    }

    delay(1000);

}

}
```

Appendix IV.

Code for road detection on C++

```
#include <unistd.h>  // time library
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <highgui.h>
#include <cv.h>
#include <cxcore.h>

using namespace cv;
int n_dil,n_ero;

int  a =100;
int  b=50;
int  c=10;
int i=0;
int min_c=35;
int max_c=60;
int n_smo=3;

void setup()
{
        cvNamedWindow("control_1");
        cvCreateTrackbar("smoothing","control_1",&n_smo,30, NULL);
        cvCreateTrackbar("dilation", "control_1", &n_dil,300, NULL);
        cvCreateTrackbar("errosion", "control_1", &n_ero,100, NULL);

        cvNamedWindow("canny");
        cvCreateTrackbar("min","canny",&min_c,300, NULL);
        cvCreateTrackbar("max", "canny", &max_c,300, NULL);


        cvNamedWindow("PHough line");
        cvCreateTrackbar("Var_1","PHough line",&a,300, NULL);
        cvCreateTrackbar("var_2", "PHough line", &b,300, NULL);
        cvCreateTrackbar("vaar_3", "PHough line", &c,300, NULL);


}

int main(int argc, char** argv)
{
        Mat src, dst, color_dst,blu,out;
        Mat less;
```

```
CvCapture* capture = cvCaptureFromCAM(0);

do{
        setup();

        Mat pic;
        pic=imread(argv[1], CV_LOAD_IMAGE_COLOR);

        src=pic;

        imshow("main", src);
        int x=225;
        int y=450;
        int width=600;
        int height=200;

        Rect roi(x, y, width, height);
        Mat image_roi = src(roi);

        imshow("fullltooo",image_roi);


Mat median;
src=image_roi;
int l=41;
        medianBlur( src, median,l)  ;
        imshow("med" ,median);

        int i=41;
        bilateralFilter ( src, blu, i, i*2, i/2 );
        src=median;

        imshow(" new", blu);
        if(cvWaitKey(33)==27)break;
        Canny( src, dst,min_c, max_c, 3 );

        imshow("not_filtered" ,dst);

        const Scalar& borderValue=morphologyDefaultBorderValue();
        int borderType=BORDER_CONSTANT;
        int iterations=1;
        Point anchor=Point(-1,-1);
        Mat kernel = Mat::ones(Size(4, 4), CV_8U);


        imshow("morpho",dst);

        cvtColor( dst, color_dst, CV_GRAY2BGR );
```

```
        erode(color_dst, color_dst, kernel,anchor,n_ero ,borderType,borderValue);
        dilate(color_dst, color_dst, kernel,anchor,n_dil,borderType,borderValue);

        imshow("filters" , color_dst);

        vector<Vec4i> lines;  // array of 4 dimension with integer
        HoughLinesP( dst, lines, 1, CV_PI/180, a, b, c );
        for( size_t i = 0; i < lines.size(); i++ )
        {
        line( color_dst, Point(lines[i][0], lines[i][1]),
                Point(lines[i][2], lines[i][3]), Scalar(0,0,255), 3, 8 );
        }

        imshow("find",color_dst);


        if(cvWaitKey(33)==27)break;
}while(1);

cvDestroyWindow("CV");
return 0 ;

}
```