Hexapod Modelling, Path Planning, and Control

Thesis · May 2017

CITATIONS
0

READS
456

1 author:

Canberk Suat Gurel
University of Maryland, College Park
10 PUBLICATIONS **0** CITATIONS

SEE PROFILE

MANCHESTER 1824
The University of Manchester

Hexapod Modelling, Path Planning, and Control

Third Year Individual Project – Final Report

May 2017

Canberk Suat Gurel

9023527

Supervisor: Dr Joaquin Carrasco Gomez

1. Abstract

As a result of exponentially developing science and technology, tremendous amount of research is being undertaken on mobile robots which are commonly used in both domestic and industrial applications. Hexapod robot is a type of legged mobile robot that is autonomous and omni-directional.

This report presents the hierarchical control architecture that is developed to control a six-legged robot which was modelled within the MapleSim environment. In addition, 2 different map-based path planning algorithms, A* (star) and Probabilistic Roadmap Method (PRM) are implemented to find the shortest path to a given goal and to step over obstacles, respectively. Static and dynamic analyses of the hexapod model are given and an adaptive gait selector which was implemented to improve the stability depending on the terrain slope, is presented. A geometric approach to tilt control has been developed and implemented. Moreover, a multivariable control was implemented in the low level controller of the hierarchy for position control and trajectory tracking. MATLAB and MapleSim are widely used throughout the project and the important codes are provided in the Appendix.

The obtained results show that the hexapod robot successfully navigates between point A and point B in both of the implemented approaches: slalom and pedestrian-lane-change.

Key words: Hexapod, MapleSim, Hierarchical Control, Path Planning

Contents

Final Word Count: 13,683 words

2. Introduction

The wheel was used for transportation for the first time in Mesopotamia around 3200 B.C. and even today the modern transportation relies on wheels [1]. Nevertheless, according to the US Army, it is estimated that half of Earth's surface cannot be accessed by wheeled locomotion [2]. On the other hand, legged locomotion is more advantageous in terms of mobility and overcoming obstacles, hence legged animals and humans are capable of accessing the majority of Earth's surface [3]. For this reason legged robots have been the focus of countless researches in the last two decades [4]. The legged robots are commonly used by military to transport goods over uneven terrain, nuclear power plant inspection, and in search and rescue (SAR) missions [2]. Furthermore, some legged robot are developed for the exploration and mapping of unknown environments, demining tasks, and agricultural tasks [2].

The legged robots are categorised depending on their leg types, leg orientations and joint configurations. Figure 2.1 shows the three hexapod models that are created in MapleSim. In light of the analysis given in the progress report, circular body-shaped model, shown in Figure 2.1-c was chosen for further development.

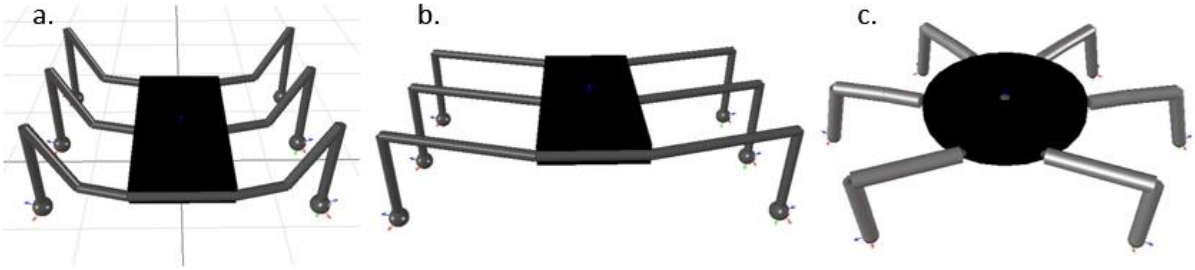


Figure 2.1 (a) Arachnid – Frontal, (b) Reptile – Frontal, (c) Reptile - Circular

The structure of this report is organized as follows; in Chapter 2, a comparison between legged and wheeled robots is provided, in Chapter 3, the fundamental concepts associated with hexapod robots are given, following that, a hierarchical control architecture that may be divided as path planning and control is given in Chapter 4. Then, the simulation results that are obtained from MapleSim are presented in Chapter 5 and the report is concluded with future work and conclusion in Chapters 6 and 7, respectively.

2.1. Statements of Aims, Objectives, and Motivation

The aim of this project was to build a hexapod robot in the MapleSim simulation platform to test different path planners and control strategies for the hexapod robot.

The steps that were taken to accomplish the aim are as follows:

- Circular and rectangular body-shaped hexapod robots are created in the MapleSim environment.
- Inverse kinematic equations are derived to coordinate the legged locomotion.
- 3 different biologically inspired walking gaits, tripod, ripple, and wave are implemented in the simulations.
- Static and dynamic stability analyses of the robot are undertaken.
- Dynamics of hexapod robot is examined and equations of motion are derived.
- Forces in each leg is calculated and a tilt control method is developed.
- A* path planning algorithm is used to find the body path that traverses from the start point to the target point.
- Probabilistic Roadmap Method (PRM) is implemented to step over obstacles within the workspace of the leg.
- Adaptive gait selector and zero moment point (ZMP) planners are implemented which improved the stability of the robot during locomotion.
- Inverse dynamics is implemented to improve the trajectory tracking of the robot legs.

2.2. Literature Review

Throughout the project, numerous resources have been studied, many different methods of implementation have been encountered and some of them were utilised to improve the project. This section is divided into 3 sub-sections which review the different legged robots, methods of path planning, and control strategies.

2.2.1. Legged Robots

As it was mentioned in the Introduction, one of the applications of legged-robots is demining; Nonami [2] and his colleagues designed the 4th generation of COMET, a hydraulically actuated hexapod robot, for land mine detection and demining. Figure 2.2–a shows a picture of COMET-IV which is powered by two 720cc gasoline motors, weighs 1500 kg and is capable of reaching a walking speed of 1 km/h [2]. COMET-IV uses force control for navigation on an uneven terrain. Figure 2.2–b shows ATHLETE which is a hybrid hexapod robot that is capable of both wheeled and legged locomotion [5]. It was developed by NASA for exploration of Mars [4]. It is capable of mapping unknown environments, collecting samples with its scoop and gripper attachments, and transporting goods that weigh up to 450 kg [6].

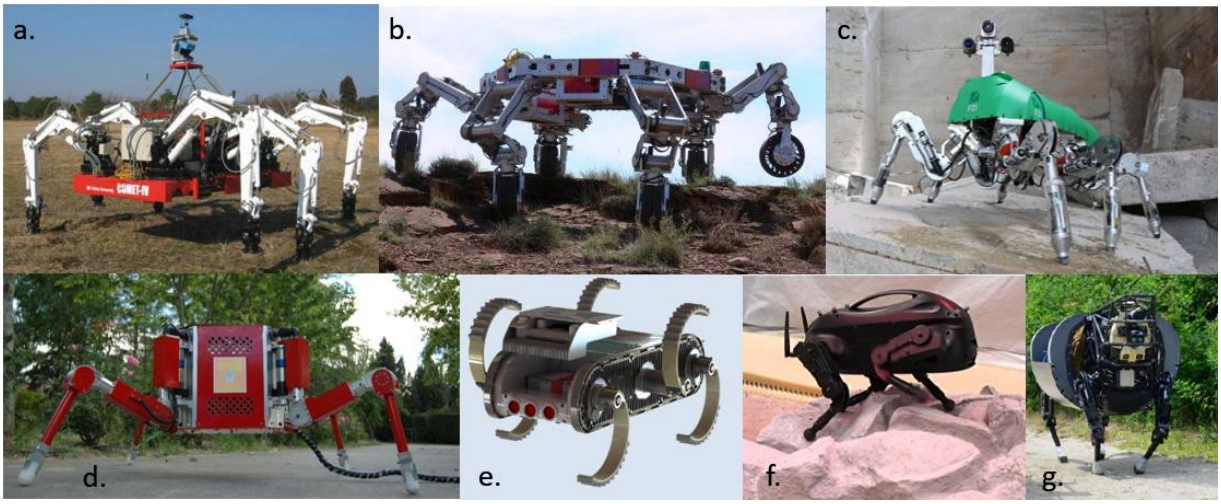


Figure 2.2 Existing legged robots, adapted from [2] [4] [7] [8]

Roennau [7] demonstrated the 5th generation of LAURON which, unlike its ancestors, is designed with 4 degrees of freedom (DOF) in each of its 6 legs, giving it advantage to climb steeper surfaces. Figure 2.2–c shows LAURON V which was designed for search and rescue (SAR) missions [7]. Figure 2.2–d shows SILO4 [3] which is a quadruped designed by Industrial Automation Institute (IAI) as an experimental testbed for artificial intelligence (AI). The design of SILO4 is provided as open-source to encourage researchers to build their own robot and to implement AI in legged robots. Figure 2.2–e shows RHex which was developed by Boston Dynamics, what makes it special is its extraordinary design. RHex is a six-legged robot that has only 1 DOF at each leg [4]. RHex weighs about 12.5 kg and it is designed specifically for rough terrain [9]. It is capable of overcoming obstacles that are larger than its size and it can operate even if it is inverted [10]. Please, refer to the progress report for the other two products of Boston Dynamics, LittleDog and BigDog which are shown in Figure 2.2-f and Figure 2.2-g, respectively.

2.2.2. Path Planning

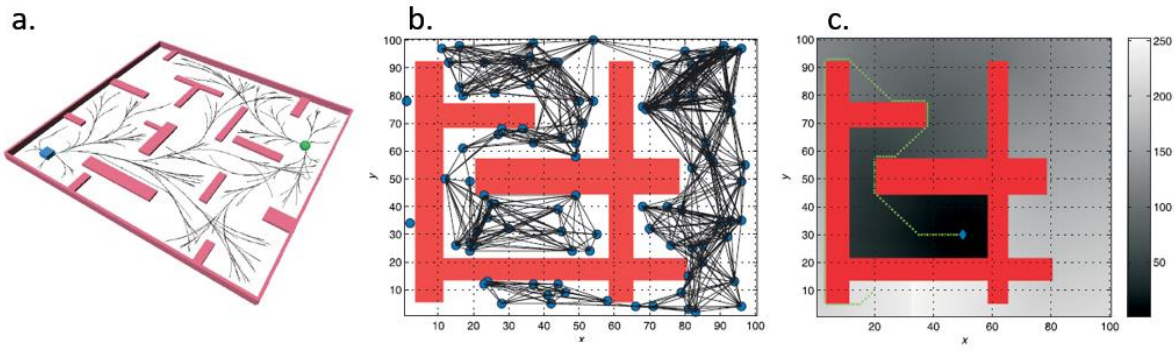


Figure 2.3 Different path planning algorithms, adapted from [11] [12]

Rapidly exploring random tree (RRT) was first introduced by LaValle [13] which is a randomised path planning method that spreads in an omni-directional manner from both start and goal points until the two trees connect with each other as it is shown in Figure 2.3–a. Belter and Skrzypczynski [14], made use of this method to obtain a collision-free trajectory over an object. Figure 2.3–b shows probabilistic roadmap method (PRM) which was used by Hauser [5] to find the shortest path between two points for a humanoid robot, HRP-2. This method is further explained in Section 4.2.1. Bai and Low [15] used artificial potential fields approach for path finding. An example of this method is shown in Figure 2.3–c where red cells represent the obstacles and grey intensity increases as the goal point is being approached. This algorithm is based on a gradient descent search method where the robot is treated as a point particle under the influence of a potential field which is attracted to the goal point and repelled from the obstacles located within the environment [16].

2.2.3. Control

The operation of robot control architectures is based on a 3 step structure that consists of sensing, response, and action [4]. Figure 2.4–a shows the reactive control architecture, where an action is defined for different sensor readings. This method is used by the RHex robot which is shown in Figure 2.2-e. Although this architecture provides rapid speed of response the method is not suitable if predictive planned outcomes need to be produced [17]. Figure 2.4–b shows the structure of hierarchical control architecture which is used by Kolter [8] and Kalakrishnan [18] to control the LittleDog robot which successfully walks over uneven terrain. This architecture is more suitable for static environments where the goal is achieved in a planned manner [17]. However in the dynamic environments, the planning phase cause a slow response to changes [17].

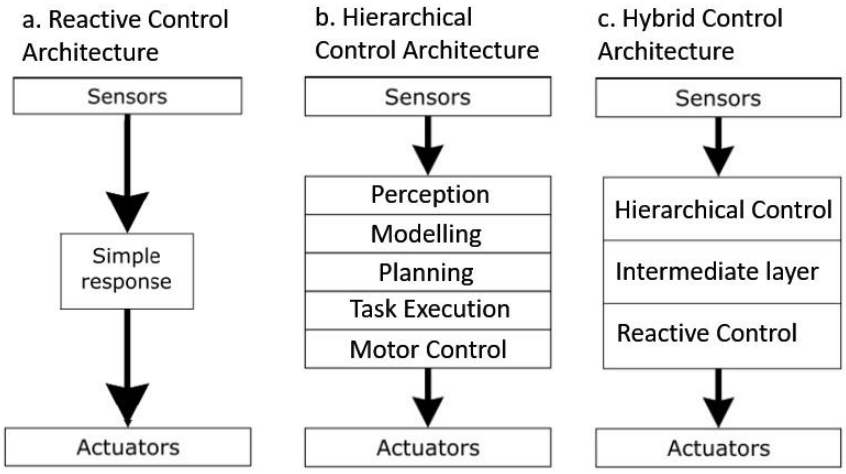


Figure 2.4 Robot control architectures, adapted from [4]

Figure 2.4–c shows the structure of hybrid control architecture where the two control architectures are combined to ameliorate the drawbacks of each architecture. The intermediate layer is required for a smooth transition between two architectures [17]. Bjelonic [19] and his colleagues implemented hybrid control architecture in hexapod robot Weaver which autonomously navigates on uneven terrain, shown in Figure 2.5.

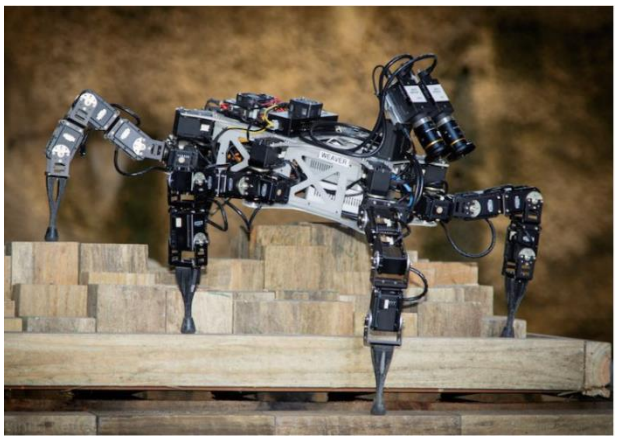


Figure 2.5 Weaver on the testbed

2.3. Advantages of Hexapod over Wheeled Vehicles

The legged robots have a number of advantages over the wheeled robots such as mobility and overcoming obstacles. Legged robots are omni-directional systems, i.e. it can change direction independently from its direction of motion whereas this is not the case for wheeled robots which require to manoeuvre in order to change direction. Figure 2.6 shows the difference of changing direction between the two types of robots, legged and wheeled robots. It can be seen that the legged robot moves side way to avoid the obstacle while the wheelled robot needs to manoeuvre.

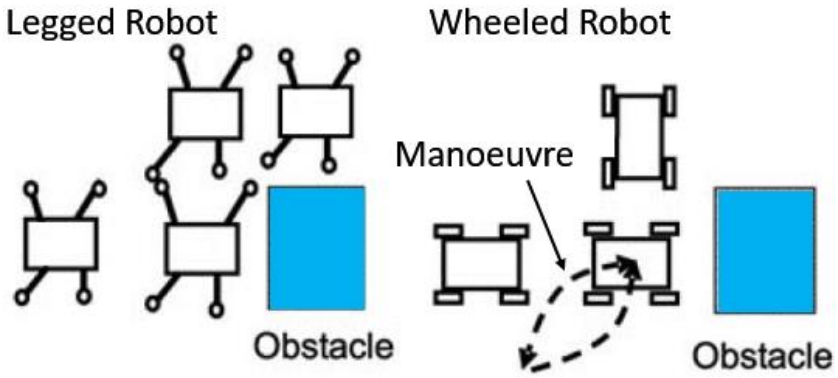


Figure 2.6 Mobility comparison of legged and wheeled robots, adapted from [3]

A legged robot is capable of maintaining its body level despite the changing slope of the surface that the robot is located on, as it is shown in Figure 2.7-a. In contrary, the body of a wheeled robot is always parallel to the slope of the surface, as it is shown in Figure 2.7-b.

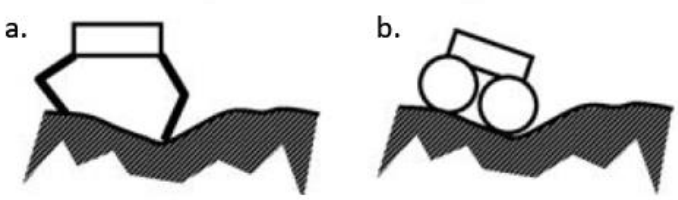


Figure 2.7 Body tiltness correction of legged robots, adapted from [3]

The legged robots are also more advantageous for tasks that require the robot to overcome obstacles and to operate on non-continuous terrain, since the legged robots are capable of stepping over an obstacle or a pit, shown in Figure 2.8-a and Figure 2.8-c, whereas a wheeled robot is simply stuck if the height of the obstacle is greater than the radius of its wheels, shown in Figure 2.8-b and Figure 2.8-d.

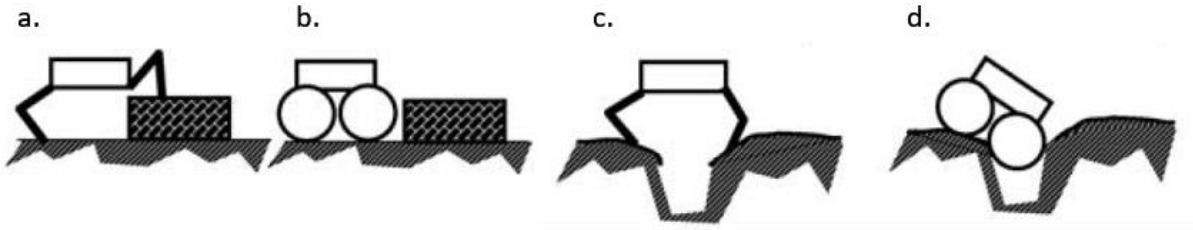


Figure 2.8 Overcoming obstacles advantage of legged robots, adapted from [3]

2.4. Disadvantages of Hexapod over Wheeled Vehicles

The legged robots are not commonly used due to the drawbacks associated with the complexity of the system, low locomotion speed, and high cost of manufacturing [3]. The legged robots are more complex in terms of mechanics and control, compared to the wheeled robots, e.g. the hexapod robot has 3 degrees of freedom (DOF) at each leg, and a total of 18-DOF whereas a 4-wheeled robot has only 2-DOF, i.e. 2 wheels for traction and steering, and 2 passive wheels [3]. In addition, a hexapod robot coordinates the motion of all 18 joints simultaneously, in order to navigate from point A to point B.

Another disadvantage of legged robots is the low locomotion speed since neither the statically stable nor the dynamically stable hexapod robots move as fast as the wheeled robots [3]. Moreover, due to the increased complexity, the legged robots require more actuators, motor drivers, and sensors, therefore the cost of manufacturing a legged robot is significantly over a wheeled robot. The Table 1-1 shows the quantity of needed components which are provided for the comparison and cost estimation purposes.

	Hexapod Robot	4 Wheeled Robot
Actuators	18	2
Motor Drivers	18	2
Controllers	18	2
Joint Sensor (Encoder)	18	2
Foot Sensor (Force)	6	0

Table 2-1 [3] Estimated figures which indicate the cost and complexity comparison

3. Fundamentals of Hexapod Robot

In this chapter, the fundamental concepts of hexapod robots such as walking gaits, stability, kinematics and dynamics are covered which are essential to obtain a better understanding of the control architecture that is delivered in the next chapter.

3.1. Walking Gaits

In this section, the phases in a leg cycle are introduced, then different gaits are given.

3.1.1. Swing and Stance phases of a Hexapod

Figure 3.1 shows the swing and stance phases of a robot leg cycle. A leg cycle is the combination of raising and placing a leg and it consists of 2 stages: the swing phase and the stance phase. During the swing phase, the leg traverses from the initial position to final position through air, shown by blue dashed line. On the other hand, during the stance phase, the leg end effector is in ground contact while the leg traverses from the final position to the initial position, moving the robot in the opposite direction to the red arrow.

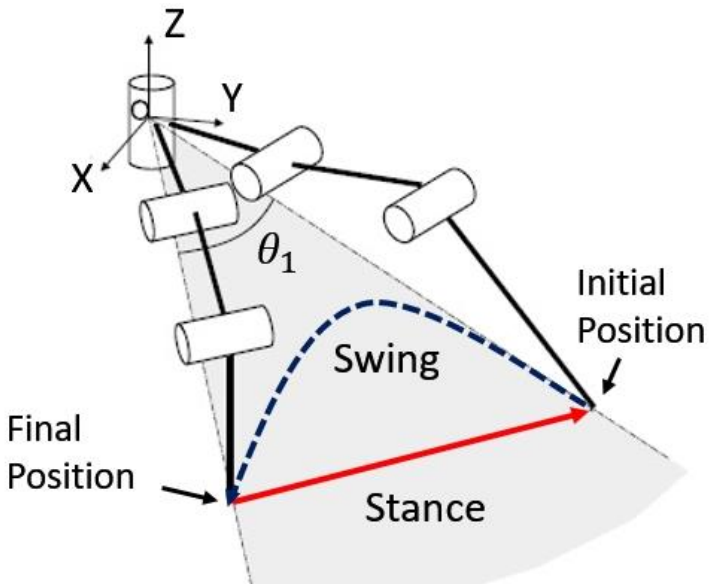


Figure 3.1 Robot leg cycle divided into swing and stance phases, adapted from [20]

3.1.2. Tripod Gait, Wave Gait, and Ripple Gait

There are 3 walking gaits that are commonly observed in the nature: wave gait, ripple gait, and tripod gait. The tripod gait was covered in the progress report which is provided in the Appendix 8.3. Wave gait is the slowest in terms of the locomotion speed, yet the most stable. In the wave gait, only 1 leg is in the swing phase at a time while the other 5 legs are in the stance phase. The locomotion speed of ripple gait is slower than tripod gait nevertheless it is faster than the wave gait, since 2 legs from opposite sides are in the swing phase by 180 degrees offset, at a time. These 2 walking gaits are shown in Figure 3.3 where the solid lines and dashed lines represent the swing phase and stance phase, respectively [17]. Figure 3.2 shows the numbering of the hexapod robot legs where the red arrow specifies the front of the robot that was created in the MapleSim environment.

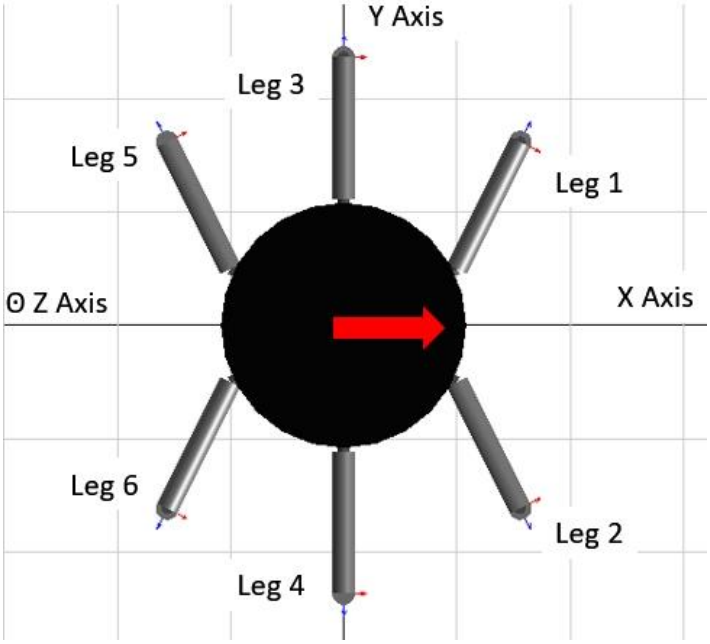


Figure 3.2: Top view of the robot model that was used in the simulations

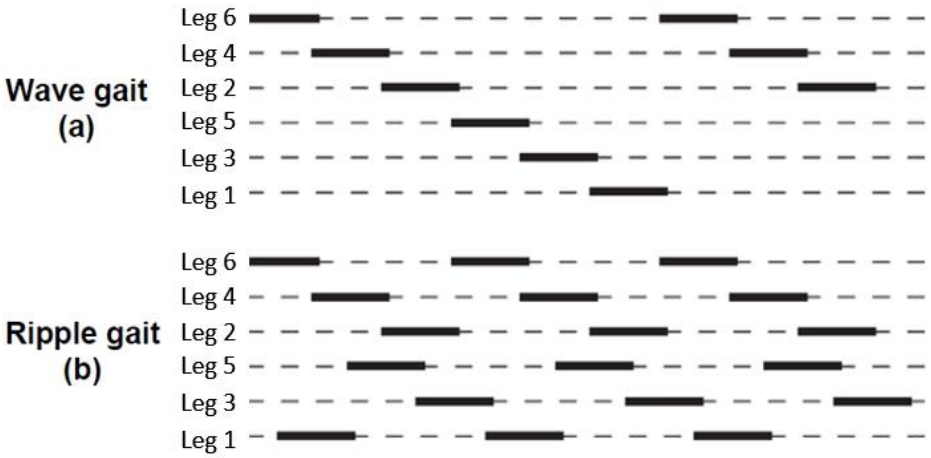


Figure 3.3 Sequence of swing and stance phases in walking gaits, adapted from [17]

3.2. Stability in Hexapod Robots

The stability of legged robots is divided into two categories: static stability and dynamic stability. To be considered statically stable, the robot needs to be stable during its entire gait cycle, without the requirement of any force to balance the robot [20]. While the robot is statically stable, the vertical projection of its centre of mass (COM) is located within the support polygon which is formed between the legs that are in the stance phase. In the case of COM being positioned on the border or outside the support polygon, the robot falls over unless it is dynamically stable, i.e. robot is balanced while walking due to the inertia caused by the motion and is statically unstable when it stops moving [21]. Figure 3.4 demonstrates the support polygon of the MapleSim model during tripod gait where the red circle denotes the vertical projection of its COM.

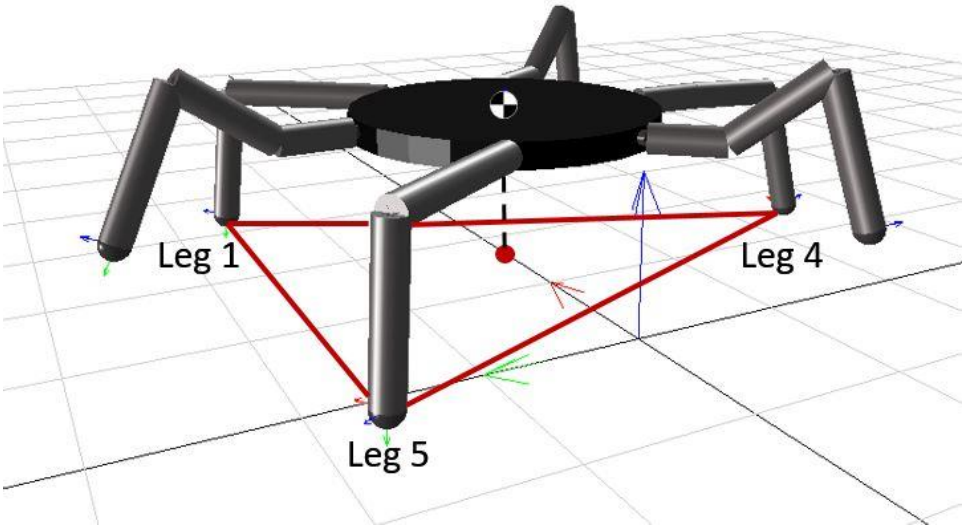


Figure 3.4 The Support Polygon that is formed During Tripod Gait.

Figure 3.5 shows the bottom view of the hexapod robot where the support polygon is decomposed into 3 sub-triangles in order to calculate the Stability Margin.

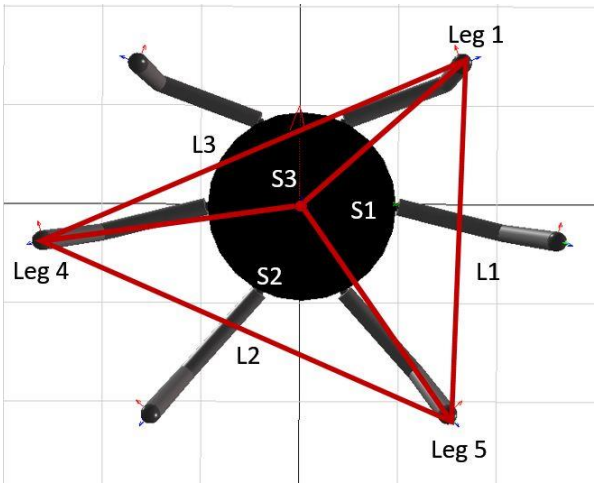


Figure 3.5 The Bottom View of the Support Polygon Decomposed into Sub-triangles.

Following [22], the area of the convex pattern between Leg 1, Leg 5 and the projection of COM is calculated by Equation 1.

$$S1 = \frac{1}{2}|(X1 - Xc)(Y5 - Yc) - (X5 - Xc)(Y1 - Yc)| \quad (1)$$

The distance between two ground contact points of Legs 1 and 5, is calculated by Equation 2.

$$|L1| = \sqrt{(X5 - X1)^2 + (Y5 - Y1)^2} \quad (2)$$

$$h1 = 2\frac{S1}{|L1|} \quad (3)$$

Since the area of a triangle is the product of base and height divided by two, the perpendicular distance, $h1$, from the vertical projection of COM to L1 is given by Equation 3. Figure 3.6 shows the heights of the 3 sub-triangles.

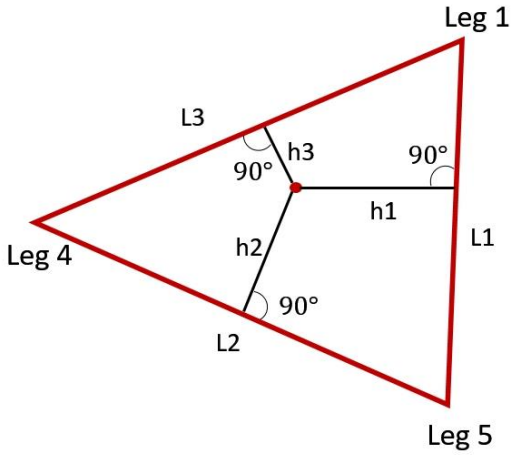


Figure 3.6 Heigths of the sub-triangles

Then, the same process is repeated to calculate S2, h2, S3 and h3. Stability margin (SM) is the shortest distance between the position of COM projection and the support polygon borders [20]. In this particular example, SM is equal to $h3$. The mathematical expression for SM is given by Equation 4.

$$SM = \min(h1, h2, h3) \quad (4)$$

The state of variable T indicates whether or not the robot is statically stable. The mathematical expression for T is given by Equation 5,

$$T = \begin{cases} 1, & \sum_{k=1}^{n} Sk = A \\ 0, & \sum_{k=1}^{n} Sk \neq A \end{cases} \quad (5)$$

where n is the number of legs in stance position at a given time and A is the area of the entire support polygon. Figure 3.7 shows the variations in SM while the robot follows the body path that is used in the experiments shown in red. The MapleSim code that was used to calculate the SM, is shown in Figure 8.1 (in Appendix).

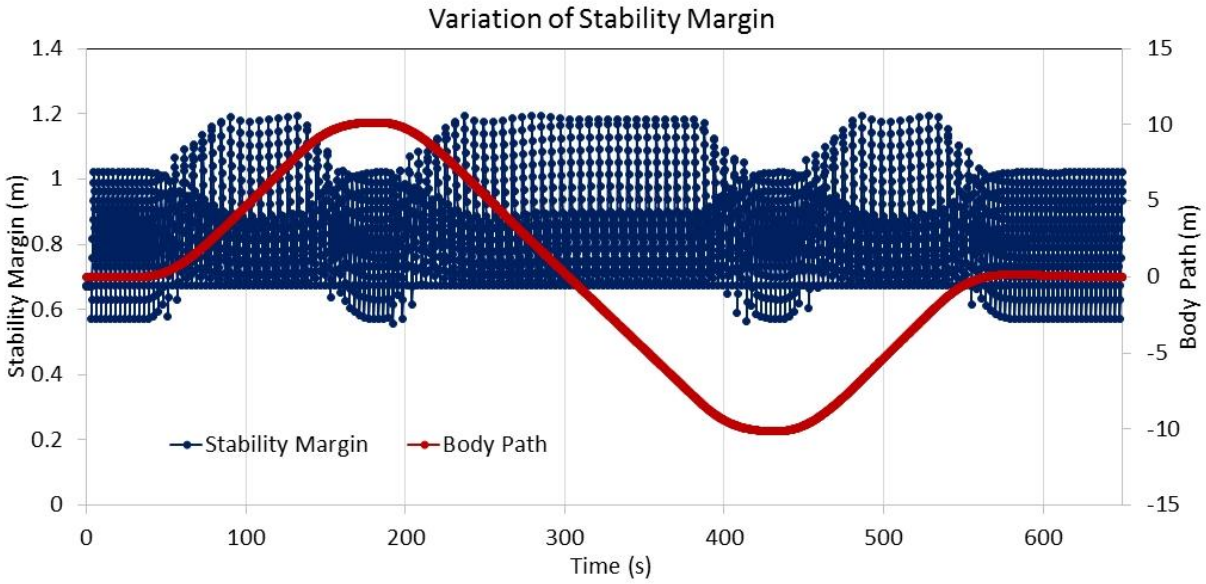


Figure 3.7 Variation in the SM while the robot is following the given body path

The variable T was used in the simulation to terminate the process if the robot becomes statically unstable. Although, this is not a concern in the level terrain, when the robot navigates on an incline surface as it is demonstrated in Figure 3.8, the vertical projection of the COM is closer to the borders of the support polygon, making the robot more likely to be statically unstable. There are two ways to solve this problem. The first one is to implement an adaptive gait generation and the second one is to make the robot dynamically stable. These two proposed solutions are tested in the MapleSim environment and the findings are presented in Section 4.2.2.

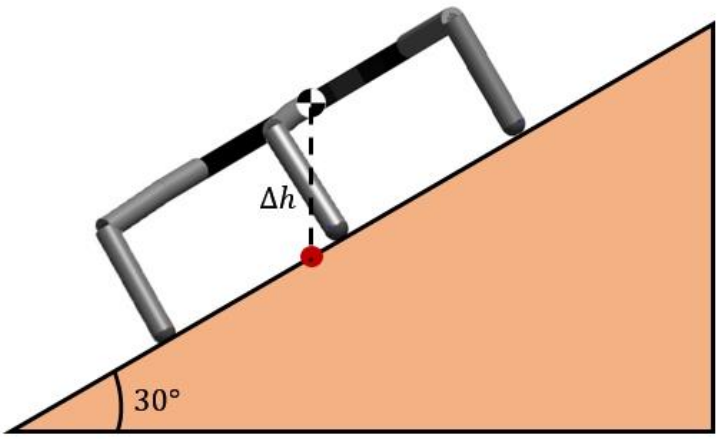


Figure 3.8: Side view of the hexapod model on an incline surface of 30°

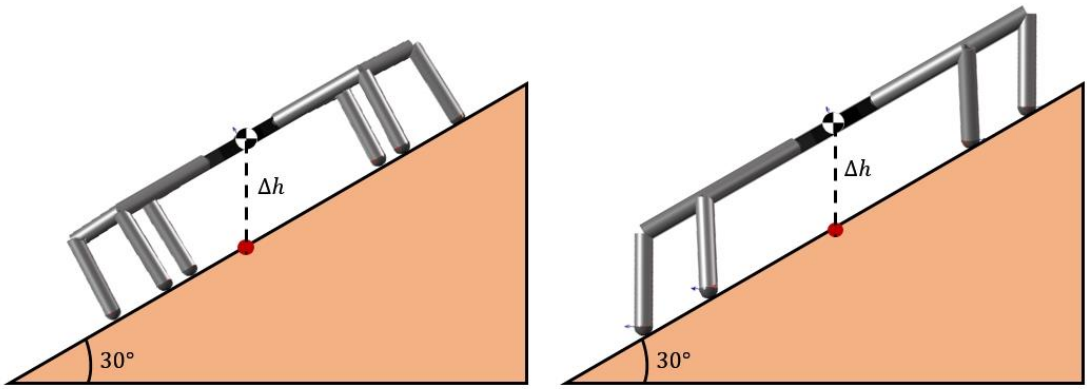


Figure 3.9: Front view of the hexapod model on an incline surface of 30°

Figure 3.9 shows how $\theta 3$ of Joint 3 is altered to ensure that Link 3 is perpendicular to the horizon, this is achieved by an geometric approach where the roll angle of the body was measured by an Euler Sensor. Then, the output from the sensor, γ was added to or subtracted from $\theta 3$ that was calculated by the inverse kinematic equations, depending on the direction of the robot. The block diagram of the procedure is shown in Figure 3.10. An analytical approach can be found in [23].

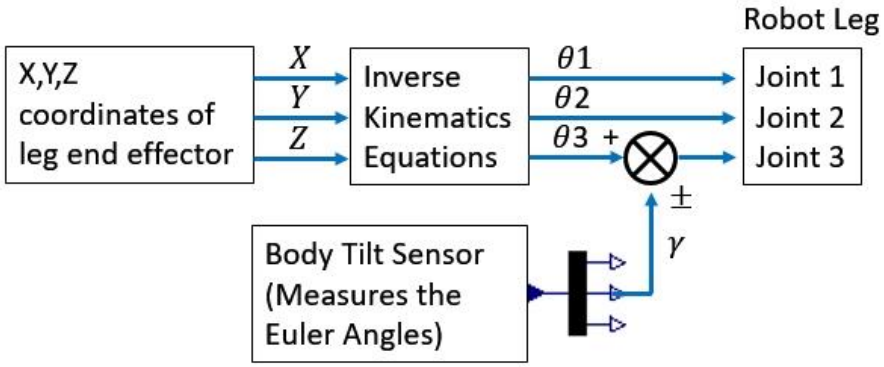


Figure 3.10: Posture Control – Block Diagram.

3.3. Kinematics of Hexapod Robot

3.3.1. Forward Kinematics

The forward kinematics is used to derive a set of kinematic equations that yields the position and the orientation of the robot end effector for the given joint parameters of the robot manipulator [16]. The derived equations were used by the footstep planner to compute the workspace of the robot leg and to project potential final configurations for the robot end effector. Denavit-Hartenberg (DH) convention states that each homogeneous transformation, H_i, can be expressed as a product of 4 transformations, given by Equation 6,

$$H_i = R(Z_i, \theta_i)\, T(Z_i, d_i)\, T(X_i, a_i)\, R(X_i, \alpha_i) \tag{6}$$

where R and T denote rotation and translation, respectively [16]. Equation 7 shows the matrix that has been calculated from the product of 4 transformation matrices given in Equation 6.

$$H_i^{i-1}(\theta_i, d_i, a_i, \alpha_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Equation 7 can be expressed in the matrix form which is given by Equation 8.

$$H_i^{i-1}(\theta_i, d_i, a_i, \alpha_i) = \begin{bmatrix} R_i^{i-1} & d_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{8}$$

The homogeneous transformation between the fixed frame and the end effector for a 3 DOF robot manipulator, H_3^0 is calculated by multiplying the transformation matrices between each joint, H_1^0, H_2^1, and H_3^2, given by Equation 9 [24].

$$H_3^0 = H_1^0\, H_2^1\, H_3^2 \tag{9}$$

Figure 3.11 demonstrates the leg of simulation model that was developed in the MapleSim environment and Figure 3.12 shows the corresponding diagram that was used to obtain the DH Parameters.

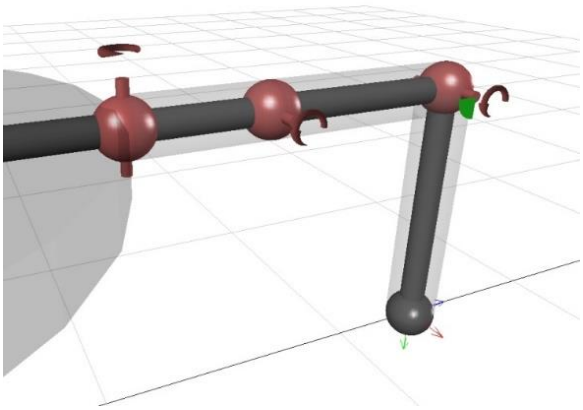


Figure 3.11: Simulation Model Leg

The DH Parameters of each leg is given by Table 2-1.

Link	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	$90°$	$L1 = 0.5\text{m}$	0	$\theta_1(t) + 90°$
2	$0°$	$L2 = 0.75\text{m}$	0	$\theta_2(t)$
3	$-90°$	$L3 = 1\text{m}$	0	$\theta_3(t) - 90°$
4	$90°$	0	0	$-90°$

Table 3-1 DH parameters

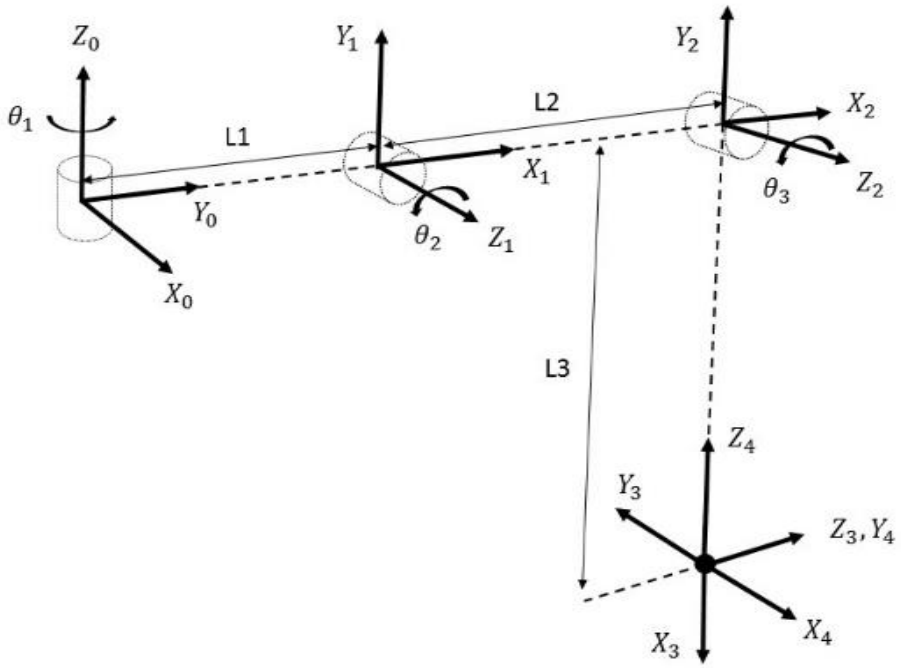


Figure 3.12 Forward kinematic model of the robot leg

In order to verify the obtained DH Parameters, MATLAB Script 8-1 (in Appendix) was used to calculate the end effector coordinates of the robot leg when it is in the rest position, i.e. $\theta_1, \theta_2,$ and θ_3 are equal to 0°. Equation 10 shows the calculated Cartesian coordinates, using MATLAB Script 8-1, which are the same as the coordinates of the end effector, shown in Figure 3.13 confirming the DH Parameters to be correct.

$$\mathrm{HT} = [0.0000 \quad 1.2500 \quad -1.0000 \quad 1.0000]^T \tag{10}$$

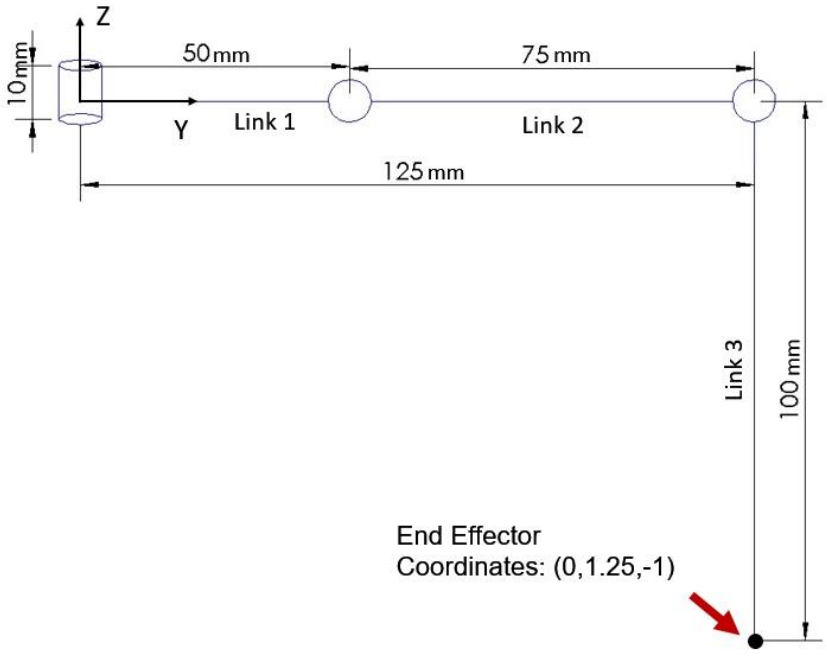


Figure 3.13 Hexapod leg in rest position

The DH Parameters that are shown in Table 2-1 are used to visualise the reachable workspace of the robot leg. Figure 3.14 illustrates the 3D view of the end effector positions and Figure 3.15 illustrates the side views where; θ_1 was altered from $-90°$ to $90°$ in increments of $10°$, θ_2 was altered from $-90°$ to $90°$ in increments of $15°$, and θ_3 was altered from $-60°$ to $135°$ in increments of $15°$. MATLAB Script 8-2 that is provided in the Appendix, was used to generate Figures 3.14 and 3.15.

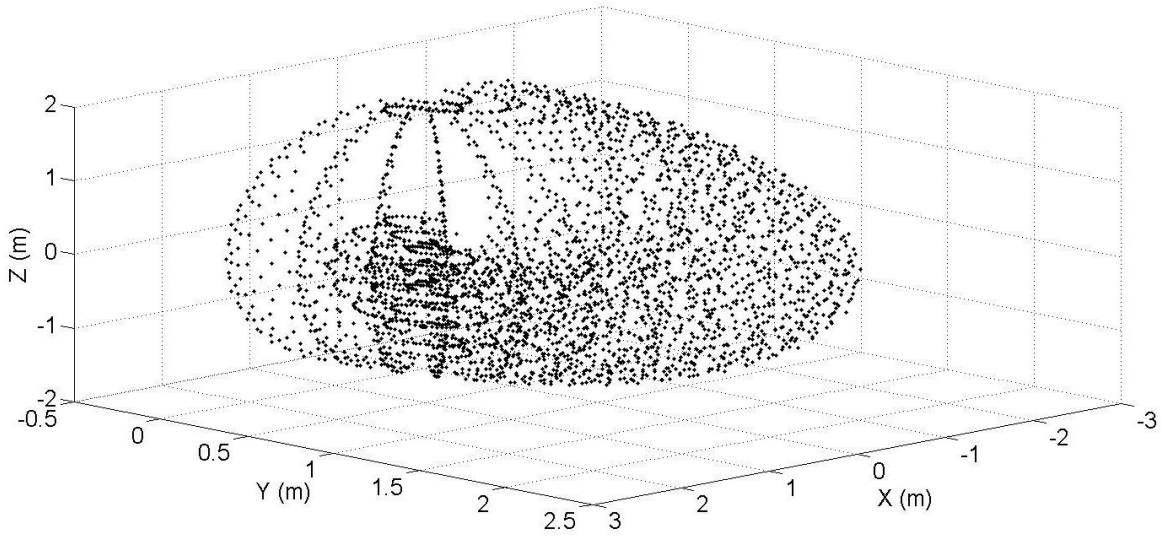


Figure 3.14 3D view of the reachable workspace of robot leg

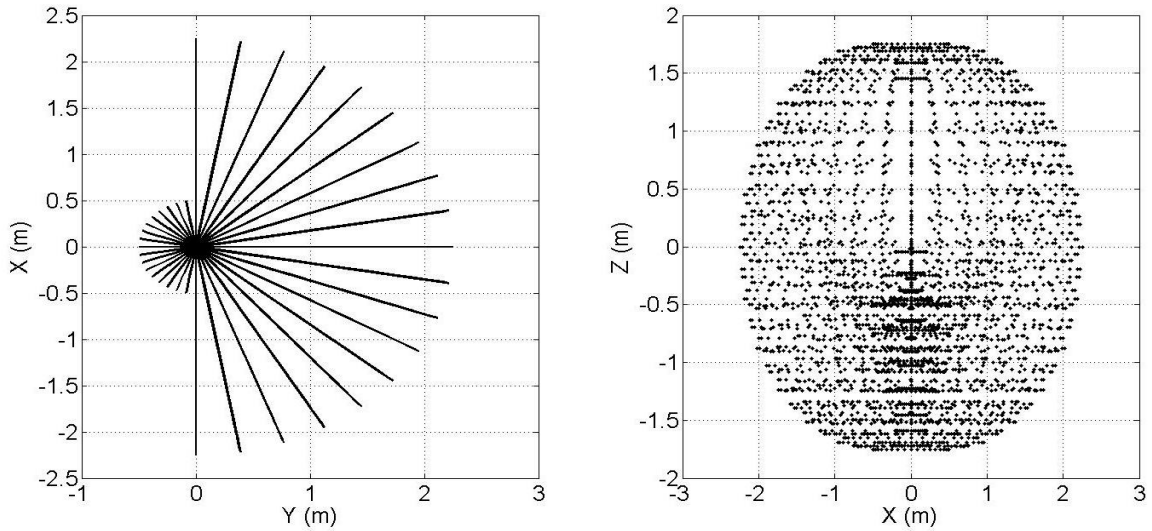


Figure 3.15 X-Y and Z-X views of the reachable workspace of robot leg

The same approach can be applied to calculate the combinations of revolute joint angles, $\theta_1, \theta_2,$ and θ_3 that cause a collision with an obstacle that is located within the workspace of the robot leg. Figure 3.17 demonstrates the angles combinations that the leg end effector collides with the obstacle shown in Figure 3.16.

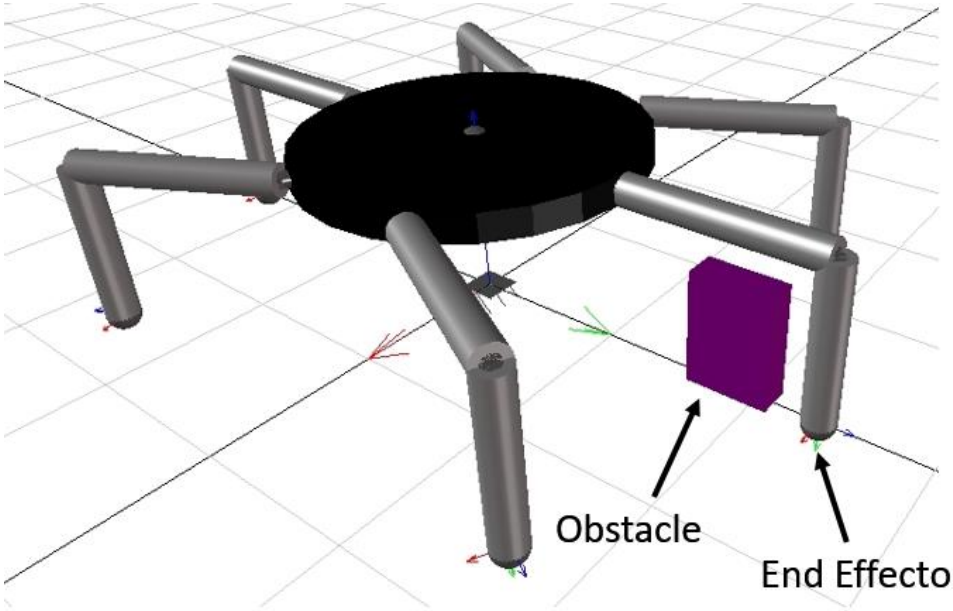


Figure 3.16 MapleSim Model and Obstacle located within the Workspace of Leg 3.

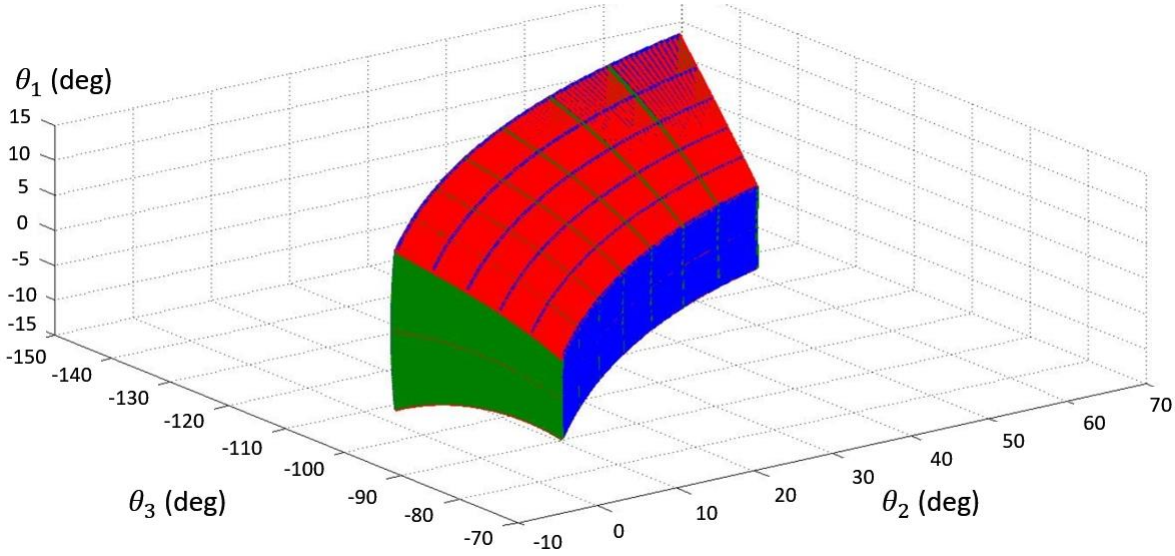


Figure 3.17 The combination of angles that cause a collision with the Obstacle.

3.3.2. Inverse Kinematics

Please refer to the progress report which is provided in Appendix 8.4.

3.4. Dynamics of Hexapod Robot

The robot dynamics provides the relationships between the robot motion, i.e. acceleration and trajectory, and the forces that are involved within the motion such as actuation and ground contact forces [3] [25]. A robot leg can be treated as a 3 DOF manipulator where the ground contact is the end effector and hence the dynamic model of a robot leg is obtained by examining the dynamics of leg mechanism [3].

3.4.1. Dynamic Model of the Leg Mechanism

The dynamic model of the leg mechanism yields a set of differential equations, i.e. equation of motion which is derived by using the Lagrangian approach [25]. The Lagrange-Euler formulation provides a relationship between the motion of the leg and the joint actuator forces [25]. Alternatively, Newton-Euler formulation may be used which, unlike Largangain formulation, requires to compute the accelerations in all $X, Y,$ and Z axis and to solve for the constraint forces [26]. The Lagrange-Euler formulation has been chosen over standard Newton formulation for deriving the dynamic equations of motion since it avoids the computation of some constraints [3] [26] [24]. Figure 3.18 shows the Denavit-Hartenberg representation of a 3 DOF robot leg where the centre of gravities (COG) of Links 1, 2, and 3 are demonstrated.

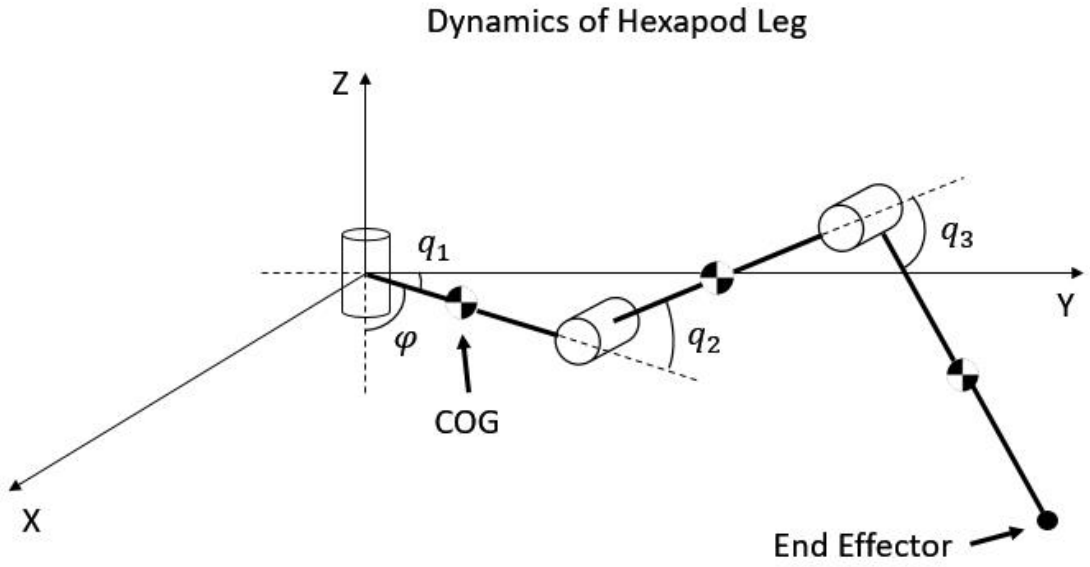


Figure 3.18 Dyanmic model of a robot leg

Since the revolute joint 1 rotates only about the Z axis, the angle, φ is equal to $\frac{\pi}{2}$ regardless of the orientation of the leg. The coordinates of each COG are calculated as given by Equations 11-19.

$$X_1 = -\left(\frac{L_1}{2}\right)\sin(q_1) \tag{11}$$

$$Y_1 = \left(\frac{L_1}{2}\right)\cos(q_1) \tag{12}$$

$$Z_1 = 0 \tag{13}$$

$$X_2 = -L_1\sin(q_1) - \left(\frac{L_2}{2}\right)\cos(q_2)\sin(q_1) \tag{14}$$

$$Y_2 = L_1 \cos(q_1) + \left(\frac{L_2}{2}\right)\cos(q_2)\cos(q_1) \tag{15}$$

$$Z_2 = \left(\frac{L_2}{2}\right)\sin(q_2) \tag{16}$$

$$X_3 = -L_1 \sin(q_1) - L_2 \cos(q_2)\sin(q_1) - \left(\frac{L_3}{2}\right)\cos(q_3 + q_2)\sin(q_1) \tag{17}$$

$$Y_3 = L_1 \cos(q_1) + L_2 \cos(q_2)\cos(q_1) + \left(\frac{L_3}{2}\right)\cos(q_3 + q_2)\cos(q_1) \tag{18}$$

$$Z_3 = L_2 \sin(q_2) + \left(\frac{L_3}{2}\right)\sin(q_3 + q_2) \tag{19}$$

The MapleSim environment was used to verify Equations 11-19. Figure 3.19 demonstrates the X, Y, and Z coordinates of COG of each link while the hexapod robot walks in the tripod gait. The plots illustrate that the data calculated by using Equations 11-19 shown in blue, is almost the same as the measured data shown in orange. Note that, the slight discrepancy that is shown by the red arrow, was caused due to the fact that the calculations assume that the end effector cannot sink into the ground. However, the ground is simulated as spring-damper system in the simulation hence the end effector overshoots after the ground contact.

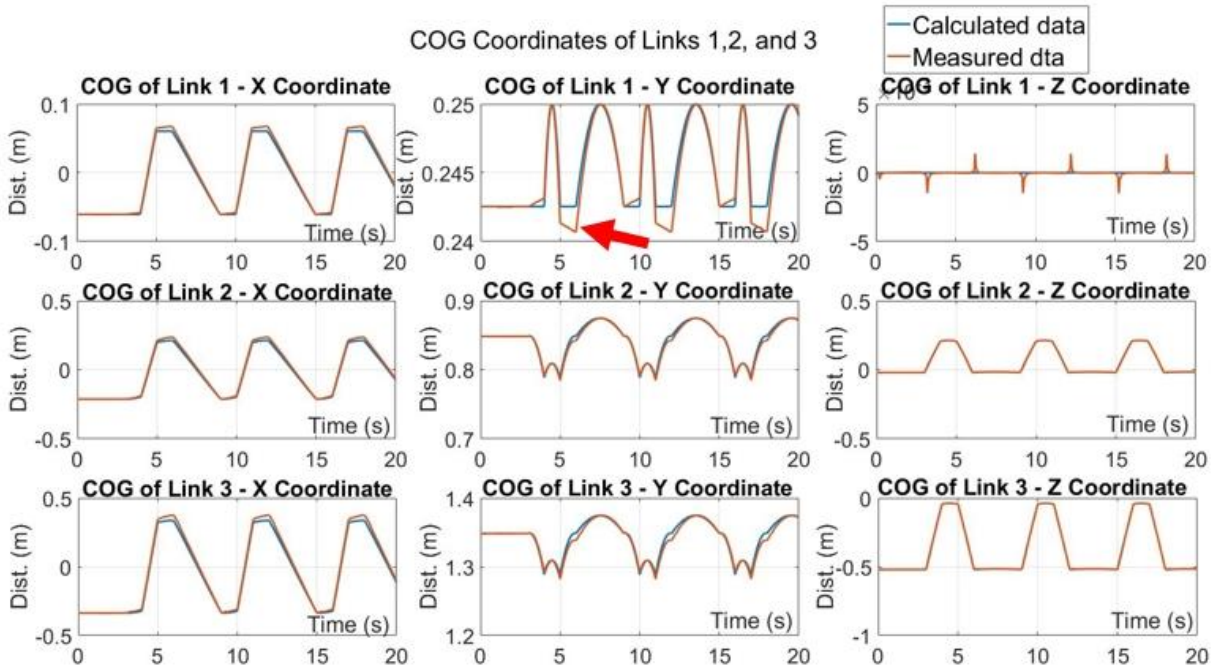


Figure 3.19 Caartesian coordinates of COG of links 1, 2, and 3

The Largangain of a mechanical system is defined by Equation 20 where T and U are the total kinetic and potential energy of the system, respectively [27].

$$L = T - U \tag{20}$$

Lagrange equation of i-th link is given by Equation 21,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i, \qquad i \in \{1,2,3\} \tag{21}$$

Equation 22 shows the dynamic expression of the robot leg in matrix form,

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau - J^T F \tag{22}$$

where $M(q)$ is the 3 by 3 inertia matrix, C is a 3 by 3 matrix such that $C\dot{q}$ is the vector of centrifugal and Coriolis terms, G is the vector of gravity terms, τ is the active joint torques, J is the Jacobian of the end effector, and F is the contact force [3] [25].

Following [16], the $M(q)$ matrix is calculated as it is given by Equation 23,

$$M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} = J_1^T \tilde{I}_1 J_1 + J_2^T \tilde{I}_2 J_2 + J_3^T \tilde{I}_3 J_3 \tag{23}$$

where the link inertia matrices, $\tilde{I}_i$, $i \in \{1,2,3\}$ are given by Equation 24,

$$\tilde{I}_i = \begin{bmatrix} \mathrm{m}_i \mathrm{I} & 0 \\ 0 & I \end{bmatrix} \tag{24}$$

where I is a 3 by 3 identity matrix, m_i is the mass of the i-th link, and moment of inertia matrix $I = \mathrm{diag}\{I_{xi}, I_{yi}, I_{zi}\}$ where I_{xi}, I_{yi}, and I_{zi} are the moments of inertia about X, Y, and Z axes of the i-th link, respectively [16].

The Jacobian matrices, J_i, $i \in \{1,2,3\}$ that are given by Equation 25,

$$J_i = \begin{bmatrix} \frac{\partial X_i}{\partial q_1} & \frac{\partial X_i}{\partial q_2} & \frac{\partial X_i}{\partial q_3} \\ \frac{\partial Y_i}{\partial q_1} & \frac{\partial Y_i}{\partial q_2} & \frac{\partial Y_i}{\partial q_3} \\ \frac{\partial Z_i}{\partial q_1} & \frac{\partial Z_i}{\partial q_2} & \frac{\partial Z_i}{\partial q_3} \end{bmatrix}, \qquad i \in \{1,2,3\} \tag{25}$$

where X_i, Y_i, and Z_i, $i \in \{1,2,3\}$ were given by Equations 11-19 [2]. The Jacobian matrices, J_1, J_2, and J_3 are computed with a symbolic software package, Maple, and the results are provided in Appendix 8.2.

Following [16] [27], the Coriolis and centrifugal forces induced on i-th joint by velocities of Joints j and k, are calculated by Equation 26,

$$C_{ij}(q,\dot{q}) = \frac{1}{2}\sum_{k=1}^{\mathrm{n}}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i}\right)\dot{q}_k\,, \qquad i \in \{1,2,3\} \tag{26}$$

The gravitational forces, $G(q)$, are given by Equation 27,

$$G(q) = \frac{\partial V}{\partial q} \tag{27}$$

where $V(q)$ is the potential energy of the COG of each link given by Equation 28,

$$V(q) = \mathrm{m}_1 g Z_1 + \mathrm{m}_2 g Z_2 + \mathrm{m}_3 g Z_3 \tag{28}$$

where m is the mass of each link, g is the gravitational acceleration, and Z is the height of the COG of each link.

3.4.2. Contact Force Estimation

Figure 3.20 shows the reaction forces, R_1, R_4, and R_5 that apply to the ground contact points, in tripod gait when legs 1, 4, and 5 are in the stance phase. The red triangle that is formed between the contact points is the support polygon of the robot. Note that, in Figure 3.20, the legs that are in the swing phase are not displayed.

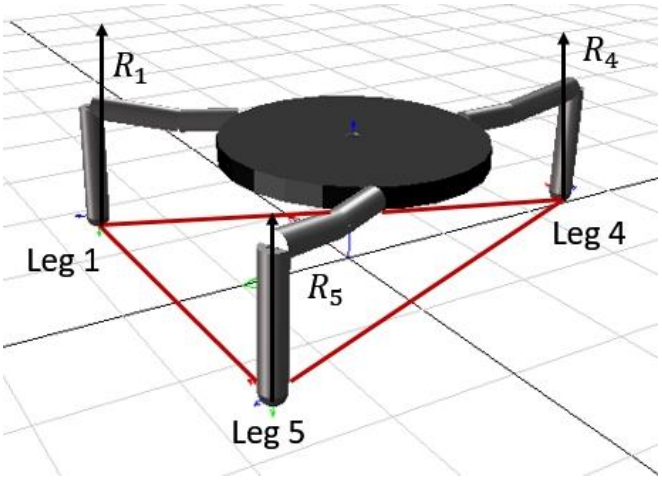


Figure 3.20 Reaction forces during tripod gait - legs 2, 3, and 6 are not shown

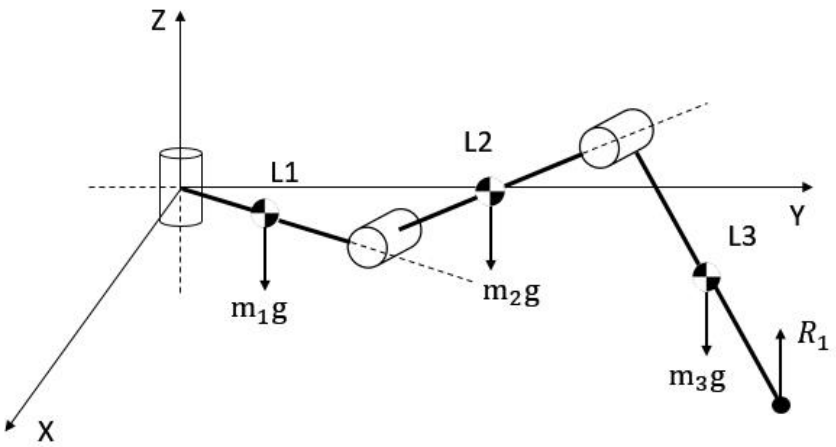


Figure 3.21 Forces associated with Leg 1

The diagram in Figure 3.21 shows the ground contact force, R_1 when Leg 1 is in the stance position, and the other 3 forces that are associated with the COG of each link. The ground contact forces in the Z direction are calculated by Equation 29,

$$R_1 + R_4 + R_5 = \left(6\sum_{i=1}^{3} \mathrm{m}_i + \mathrm{m}_c\right)\mathrm{g} + \mathrm{m}_c\ddot{Z}_c + \sum_{k=1}^{6}\left(\sum_{i=1}^{3} \mathrm{m}_i\ddot{Z}_{ki}\right) \tag{29}$$

where $\ddot{Z}_c$ is the acceleration of the body and $\ddot{Z}_{ki}$ is the acceleration of k-th leg and i-th link [28]. The other reaction forces, R_2, R_3, and R_6, when Legs 2, 3, and 6 are in the stance position are calculated with the same equation given by Equation 29.

4. Hierarchical Control Architecture

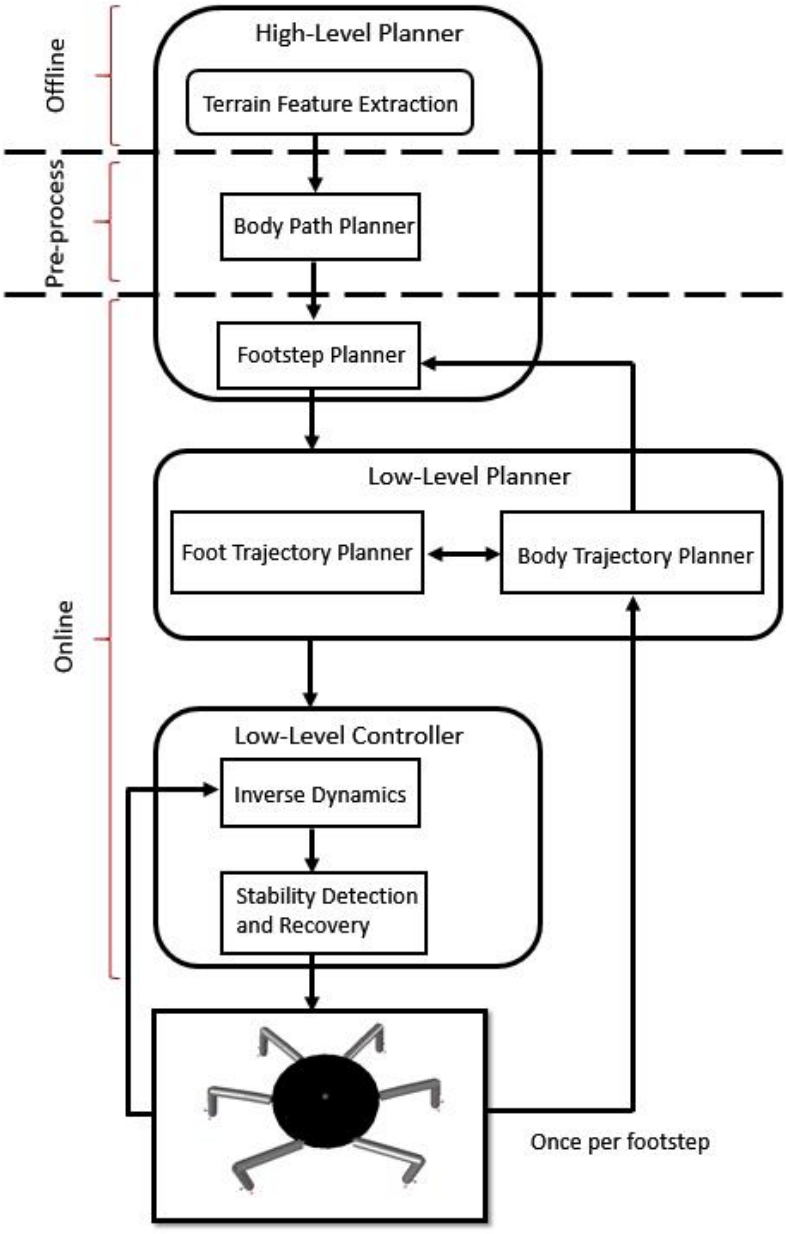


Figure 4.1 Hierarchical Control Architecture

Hierarchical control architecture operates based on the sense-plan-act robot control methodology [17]. Initially, information about the surrounding environment of the robot is gathered through the sensors, then a motion plan of the next gait cycle is calculated in a goal oriented manner. Finally, the planned action is undertaken through actuators. Then the same procedure is repeated until the goal of the given task is achieved [17]. Figure 4.1 shows the block diagram of the hierarchical control architecture that is implemented in the hexapod robot. It is assumed that the terrain features are provided before the execution of the simulation.

4.1. High Level Planner

The High Level Planner consists of the Body Path Planner and the Footstep Planner.

4.1.1. Body Path Planner

For a given start point, A and a goal point, B, the shortest distance may be calculated using a map-based planning method. Note that, the shortest distance may have a higher movement cost, i.e. there might be a longer path with fewer obstacles and easier to navigate. In order to designate whether or not this is the case, the cost map of the terrain needs to be taken into account which indicates how easy the terrain is to navigate over [8]. In this report, the shortest path is assumed to be the one with the lowest movement cost.

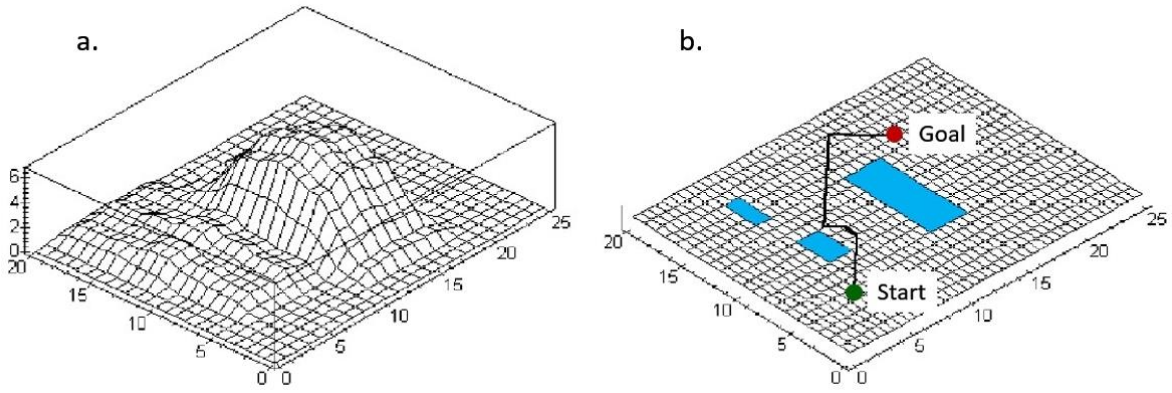


Figure 4.2 Uneven terrain and the 2D representation of terrain, adapted from [15]

Figure 4.2-a shows an uneven terrain where the regions that are higher than the reachable workspace of the robot leg, are represented as obstacles in the 2D grid that is shown on Figure 4.2-b. For example, Figure 4.3 shows a terrain height where the robot is capable of stepping over, hence in the 2D grid representation, also referred as occupancy grid representation, of this region is not an obstacle.

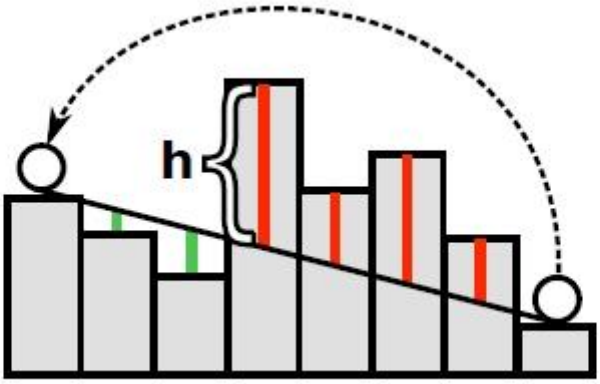


Figure 4.3 Collision map of a foot trajectory over an obstacle, adapted from [29]

The occupancy grid framework provides information about terrain regarding to the free spaces and obstacles where each grid is roughly equal to the size of the robot. [30] For the computation, the occupancy grid is represented as a multidimensional matrix where 0 denotes free space, Q_{free}, and 1 denotes obstacle, Q_{obs} [12].

The path planning algorithm, A* (star), that was implemented to compute the body path, runs in discrete space, however the real-world is continuous hence the map needs to be discretised which is done by using an approximate cell decomposition method, trapezoidal decomposition [31]. Figure 4.4 shows the occupancy grid that was used in the MapleSim simulations. Using the trapezoidal decomposition approach, it can be decomposed into 7 cells. The idea is to extend a vertical line in Q_{free} until it touches either a Q_{obs} or reaches the borders of occupancy grid [32] [33]. There is a harder-to-compute decomposition method, named as exact cell decomposition, which covers the exact shape of Q_{obs} using convex polygons, i.e. this method produces cells with irregular boundaries. However, since all the obstacles are chosen to be rectangular, trapezoidal decomposition is sufficient enough in this particular example.

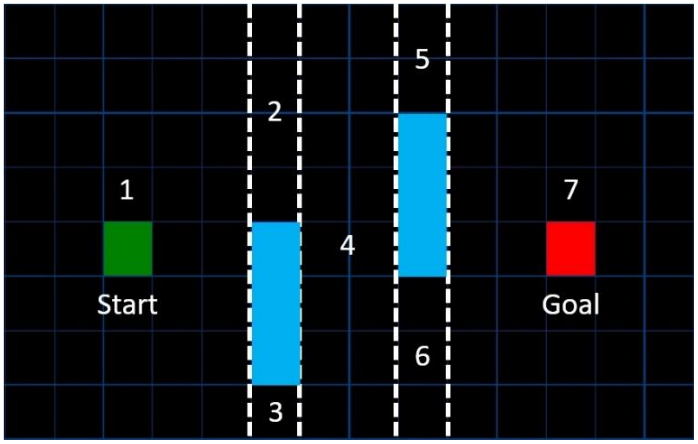
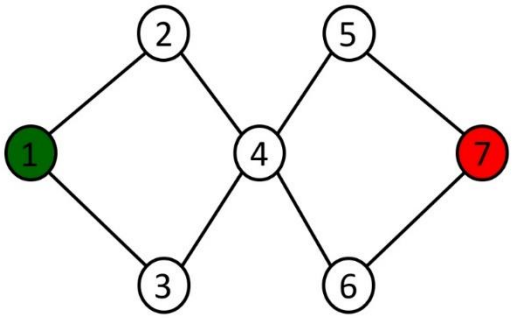


Figure 4.4 Occupancy Grid Representation



Then, an adjacency map is generated where each cell is represented as a node and the adjacent cells are connected with a line, shown in Figure 4.5. The adjacency map can be expressed in matrix form, where the adjacent cells are represented as 1 and the rest as 0, given by Equation 30.

Figure 4.5 Adjacency Map of the Occupancy Grid.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad (30)$$

A* (star) path planning algorithm determines the cost of the 8 surrounding nodes by using Equation 31,

$$F = G + H \quad (31)$$

where G is the distance from the start node and H is the estimated distance from the end node using heuristics [31]. G and H are calculated ignoring the obstacles, where the vertical and horizontal movements are counted as increments of 1 whereas the diagonal motions are counted in increments of $\sqrt{2}$. Table 3-1 shows an example of how G and H were calculated for Nodes A, B, and C that are shown on Figure 4.6. The nodes with the lowest cost, F, are given priority in the exploration process.

	Node A	Node B	Node C
G	1	2	$2+\sqrt{2}$
H	8	7	$5+\sqrt{2}$
F	9	9	9.8284

Table 4-1 Example G, H, and F values of Nodes A, B, and C.

MATLAB Script 8-3 (in Appendix) was used to generate the path in Figure 4.6 which shows the generated body path between Start node and Goal node in grey and the obstacles are shown in black.

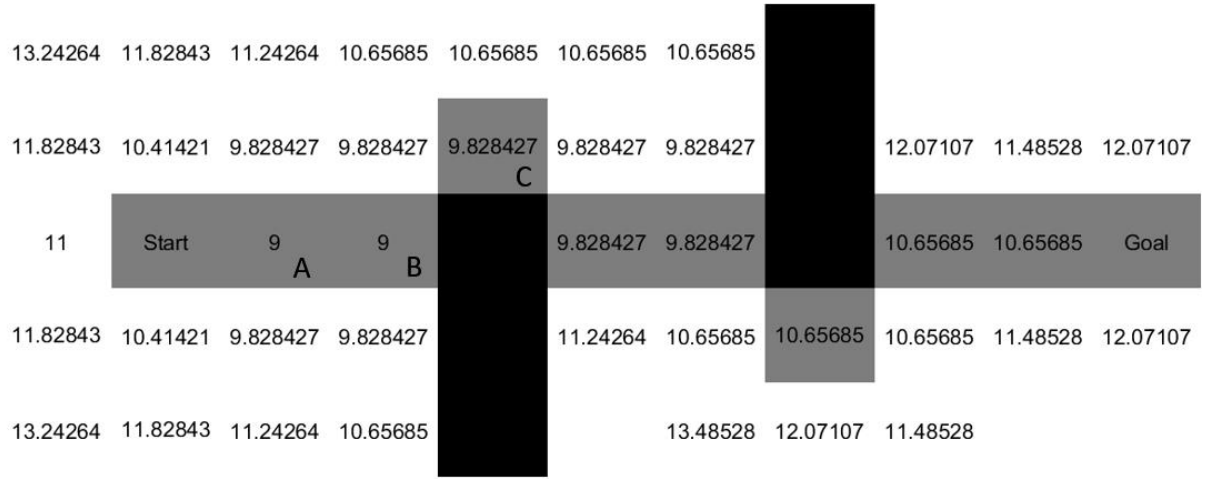


Figure 4.6 The body path found by the A* algorithm

It can be seen that, the path given in Figure 4.6 is not smooth. In order to obtain a smoother body path, a 2nd order low pass Butterworth filter is applied and hence, the body path given in Figure 4.7 is generated. This path has been used in the MapleSim simulations as the test body path.

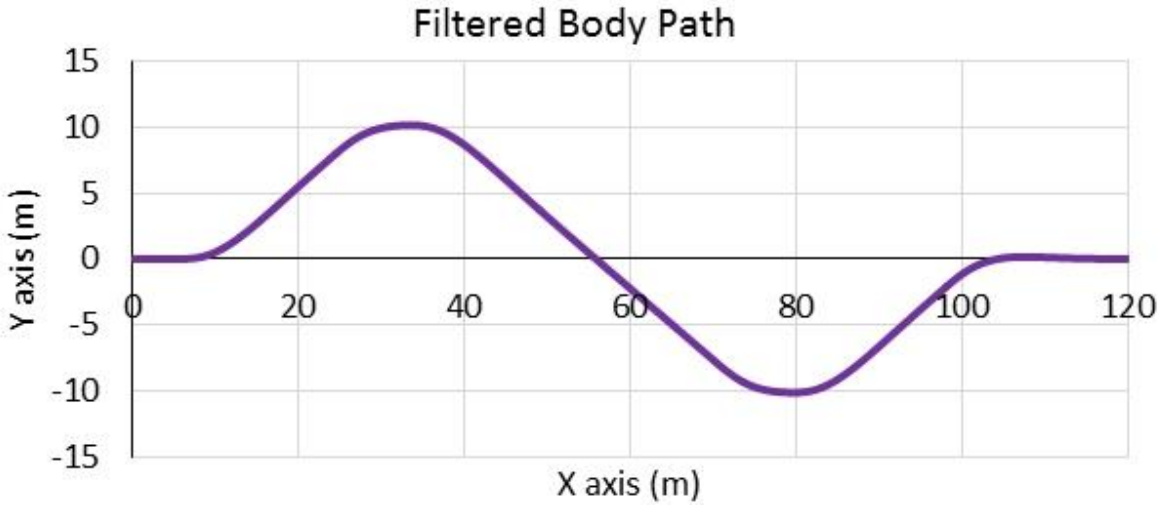


Figure 4.7 The body path that is used in the MapleSim simulations

4.1.2. Footstep Planner

The footstep planner, provides the initial configuration, q_0, and final configuration, q_f that are required by the foot trajectory planner to generate the trajectory that traverses from q_0 to q_f where q_0 is the current foot position and q_f is the position of the next footstep [16]. The footstep planner makes a projection of where the leg end effector needs to be positioned for the next step in order to follow the path that was given by the body path planner [18]. Figure 4.8 illustrates the reachable area of the leg end effector in the increments of $10°$ where the Z axis was kept constant at -1 in order to represent a level terrain.

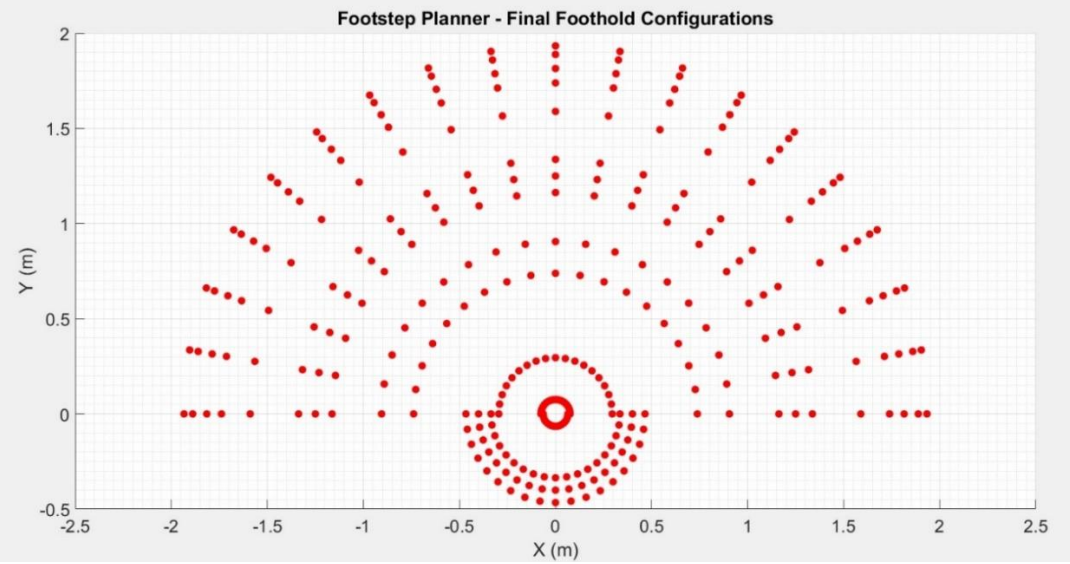


Figure 4.8 X-Y View of the possible Final Foothold Configurations.

Since it is assumed that the robot only moves forward and sideways but not backwards, the possible final foothold configurations that can be chosen by the footstep planner are limited by the area that is given by Figure 4.9.

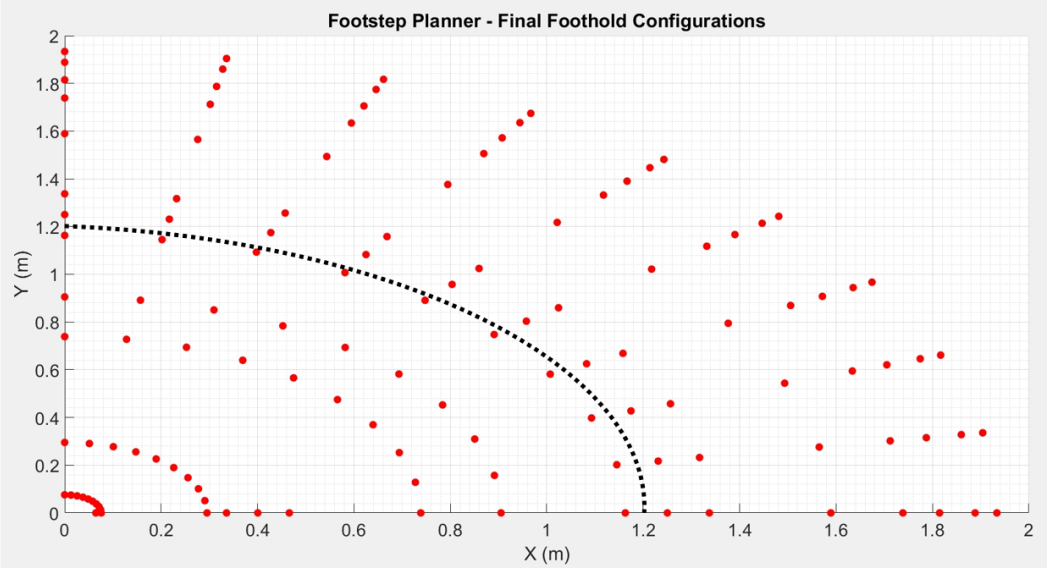


Figure 4.9 X-Y View of the possible Final Foothold Configurations in the 1st Quadrant.

The procedure to compute the final configuration, q_f is as follows. The derivative of the path signal that is given by the body path planner is computed with respect to time to obtain the slope, $\alpha(t)$ as a function of time. Then, the signal is amplified by a gain, G which was selected such that the maximum value of $\alpha(t)$ corresponds to $\frac{\pi}{2}$ where the robot moves vertically. Equations 32 and 33 are used to calculate the Cartesian coordinates, X and Y of the next final foothold configuration such that the robot follows the given body path.

$$X(t) = |\mathrm{R}\cos(\alpha(t))| \tag{32}$$

$$Y(t) = |\mathrm{R}\sin(\alpha(t))| \tag{33}$$

where R is a constant that specifies how much further the end effector will be positioned from the fixed frame of the individual leg. In the MapleSim simulations, R is chosen to be 1.2 due to the torque constraints on the joint actuators, hence the actual operational area of the leg end effector is the area that was enclosed by the black dashed line shown in Figure 4.9. Figure 4.10 shows 4 different leg configurations and the corresponding horizontal distances to the geometric centre of the body.

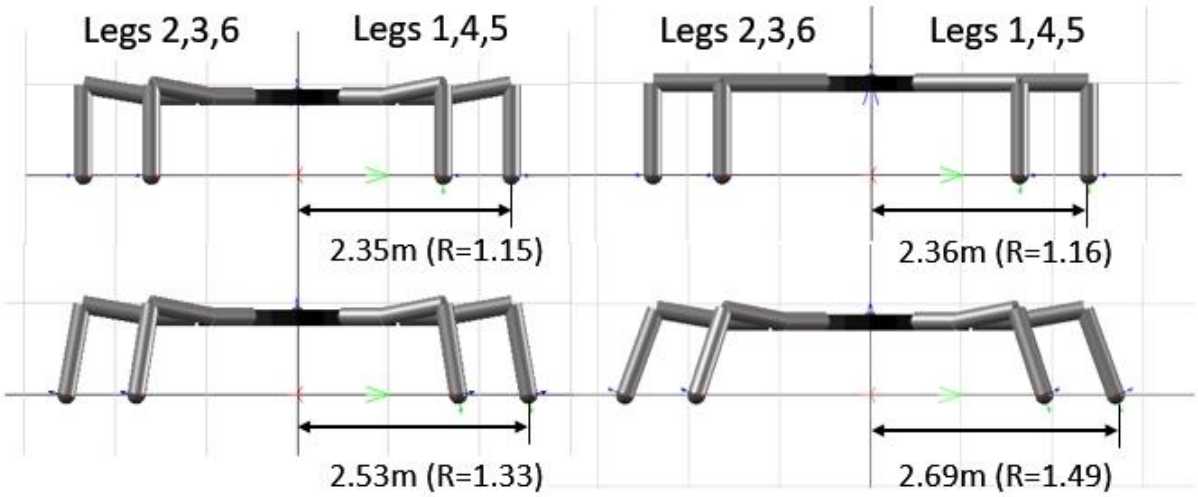


Figure 4.10 Different leg configurations and corresponding R values

The Z coordinate of the q_f is a matter of altitude which is obtained by finding the corresponding height of the point from the height map of the terrain for any given X and Y coordinates. Figure 4.12 shows the possible q_f positions on an arbitrarily generated uneven terrain such that the chosen random variables lie within ± 0.25m of the exponential function given by Figure 4.11 which represents a hill. MATLAB Script 8-4 (in Appendix) was used to create the uneven terrain that is shown in Figure 4.12.

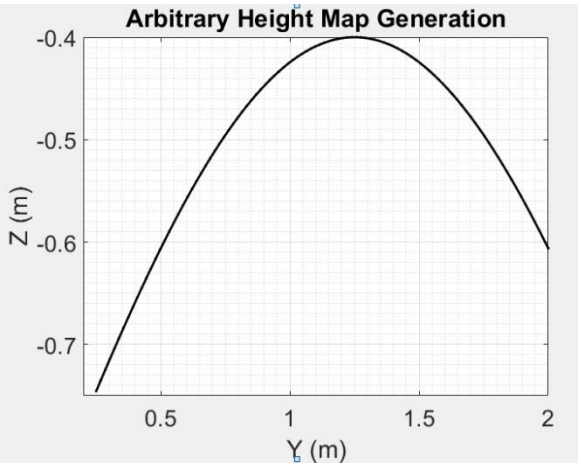


Figure 4.11 Curve that was used to generate the uneven terrain

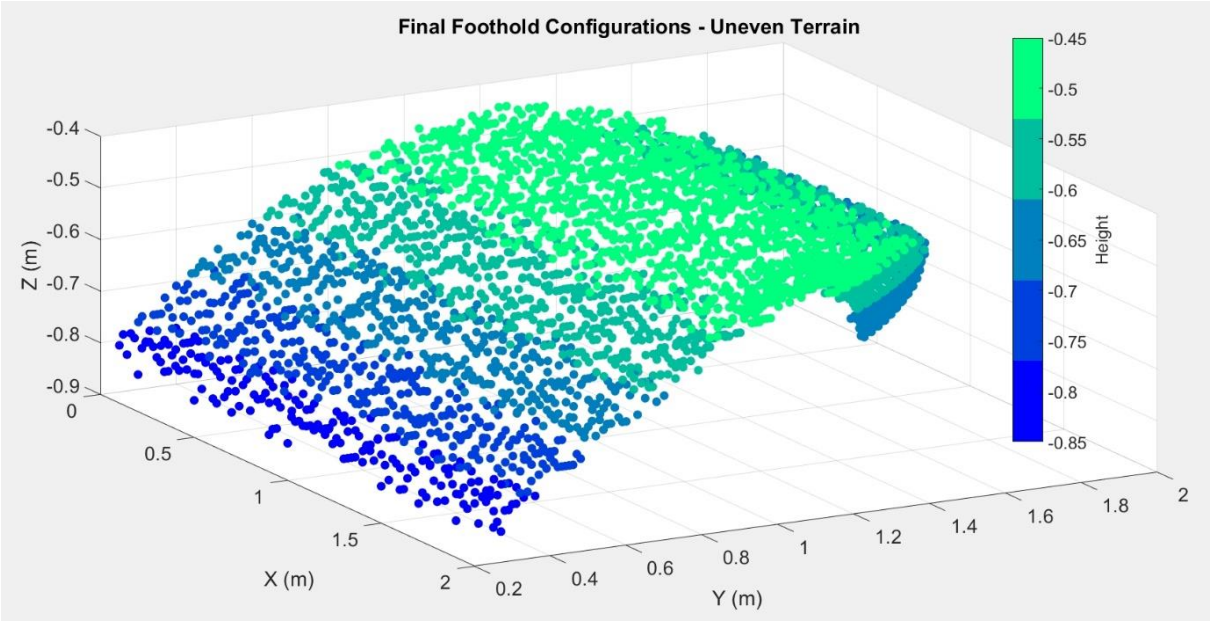


Figure 4.12 3D View of the possible final foothold configurations on an uneven terrain

Figure 4.13 shows the X-Y view of the positions of q_f on an arbitrarily generated height map, it can be seen that the X axis was incremented by 0.1m in order to reduce the computation time.

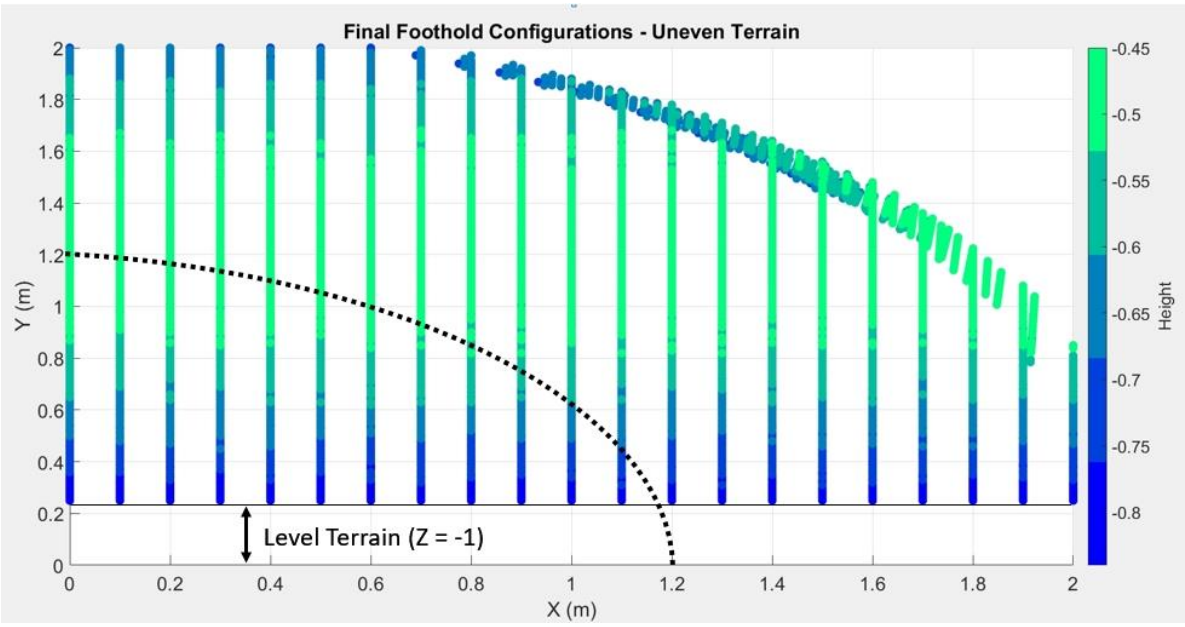


Figure 4.13 X-Y view of the final foothold configurations on an uneven terrain

4.2. Low Level Planner

4.2.1. Foot Trajectory Planner

The kinematic analysis of a robot manipulator is concerned with the geometry of the robot and the constraints due to the existence of an obstacle within the workspace of the robot leg are completely neglected. The possibility of collisions is taken into

account by the foot trajectory planner and a collision free path is generated [16].

Time, $\boldsymbol{t}$	Position, $\boldsymbol{q}$	Velocity, $\frac{\boldsymbol{dq(t_0)}}{\boldsymbol{dt}}$	Acceleration, $\frac{\boldsymbol{dq(t_f)}}{\boldsymbol{dt}}$
$\boldsymbol{t_0}$	$q(t_0) = q_0$	$v(t_0) = v_0$	$a(t_0) = a_0$
$\boldsymbol{t_f}$	$q(t_f) = q_f$	$v(t_f) = v_f$	$a(t_f) = a_f$

Table 4-2 The notations that are used in the text

Table 3-2 gives the notations that are used in this section of the report. The foot trajectory planner computes the dynamics of the leg end effector motion during the swing phase as it traverses from an initial configuration, q_0, to a given final configuration, q_f [16].

Figure 4.14 shows a foot trajectory which is decomposed into 3 sections labelled as 1, 2, and 3, respectively. During section 2, the leg moves faster whereas in sections 1 and 3, the leg moves slower to avoid an impact with the ground. Therefore, a minimum number of 4 constraints; initial configuration, q_0, final configuration, q_f, initial velocity, v_0, and final velocity, v_f, needs be taken into account.

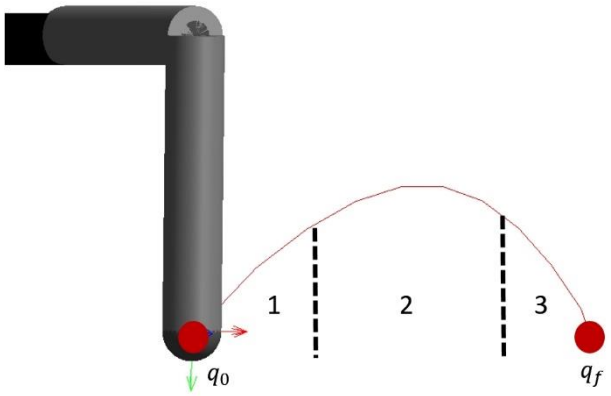


Figure 4.14 Foot Trajectory between initial and final configurations.

Control over a_0 and a_f is required to gradually accelerate and decelerate during section 1 and section 3, respectively. Consequently, quantic polynomial, $q(t)$, with 6 coefficients, needs to be used to satisfy all 6 boundary conditions on initial and final position, velocity, and acceleration [12]. Equation 34 shows the quantic polynomial where a_i, $i \in \{0,1,2,3,4,5\}$ are the coefficients of the polynomial.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (34)$$

Following [16], Equation 34 is used to derive an expression for each boundary condition as follows,

$$q_0 = a_0 + a_1 t_0 + a_2 {t_0}^2 + a_3 {t_0}^3 + a_4 {t_0}^4 + a_5 {t_0}^5 \quad (35)$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 {t_0}^2 + 4a_4 {t_0}^3 + 5a_5 {t_0}^4 \quad (36)$$

$$a_0 = 2a_2 + 6a_3 t_0 + 12a_4 {t_0}^2 + 20a_5 {t_0}^3 \quad (37)$$

$$q_f = a_0 + a_1 t_f + a_2 {t_f}^2 + a_3 {t_f}^3 + a_4 {t_f}^4 + a_5 {t_f}^5 \quad (38)$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 {t_f}^2 + 4a_4 {t_f}^3 + 5a_5 {t_f}^4 \quad (39)$$

$$a_0 = 2a_2 + 6a_3 t_0 + 12a_4 {t_0}^2 + 20a_5 {t_0}^3 \quad (40)$$

MATLAB Script 8-5 (in Appendix), was used to calculate the position, velocity and acceleration of each joint of Leg 1. Figure 4.15 demonstrates the variation in position, velocity and acceleration of Joints 1, 2, and 3 during a complete leg cycle of 4 seconds.

The velocity plots, in Figure 4.15, show that velocity of joints peaks at the centre of each one-second segment. In other words, for the majority of the time, the velocity is below its maximum value which is inefficient and hence undesirable. In order to solve this problem, linear segment with parabolic blend (LSPB) method was implemented which has increased the duration for which the joint operates at its maximum velocity [12]. Figure 4.16 shows the position, velocity and acceleration profiles calculated by LSPB method and the code is given in MATLAB Script 8-6 (in Appendix). It can be seen that the period of time that the joint operates at maximum velocity has increased.

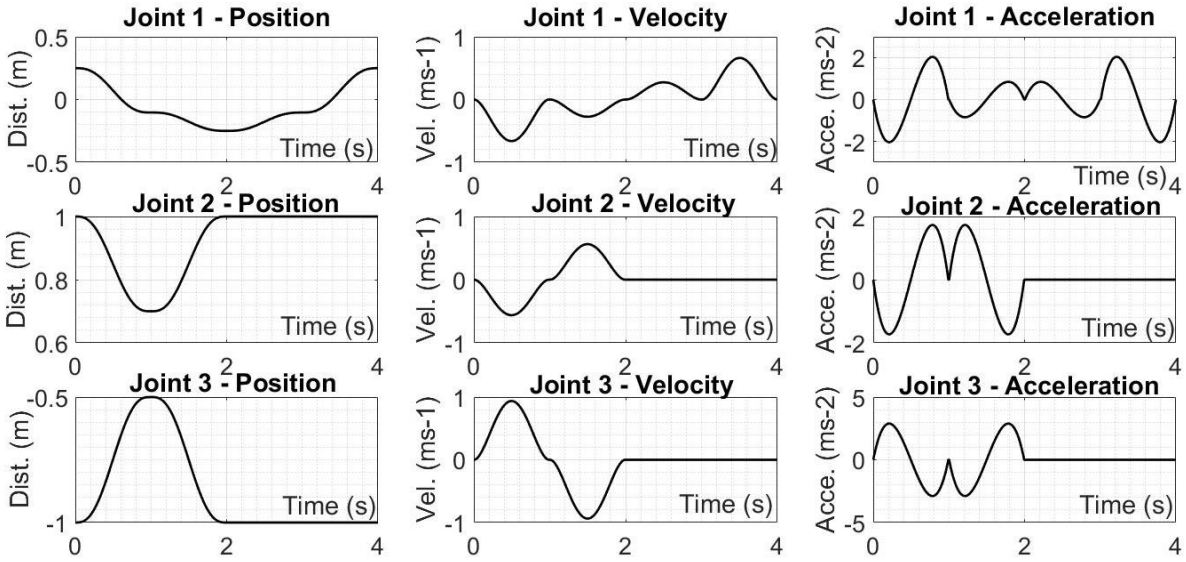


Figure 4.15 Position, Velocity, and Acceleration profiles for Quintic Polynomial

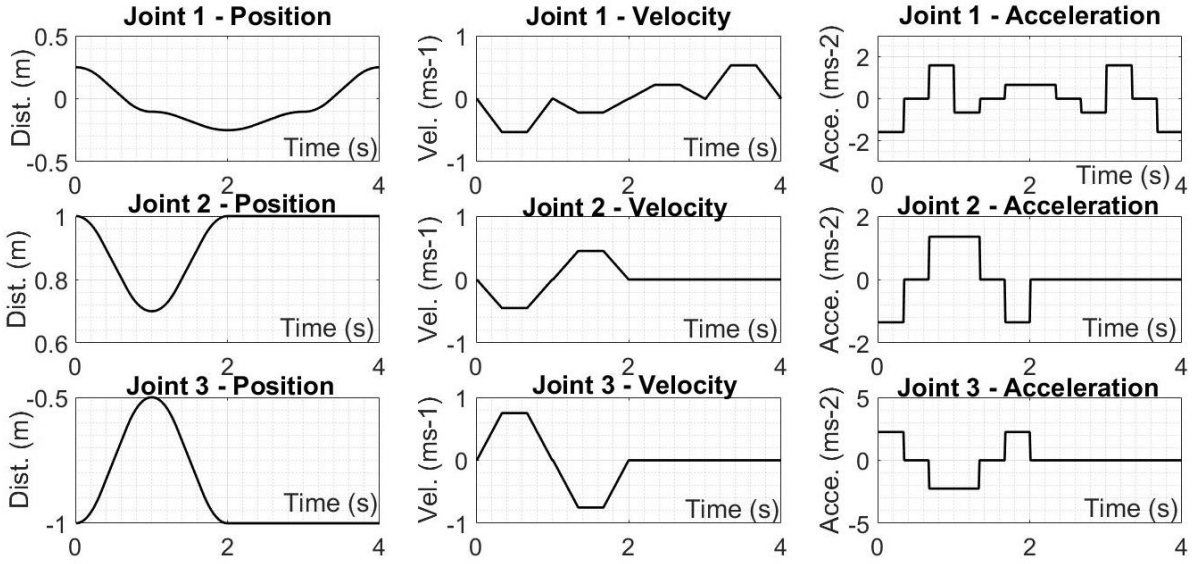


Figure 4.16 Position, Velocity, and Acceleration calculated by the LSPB Planner

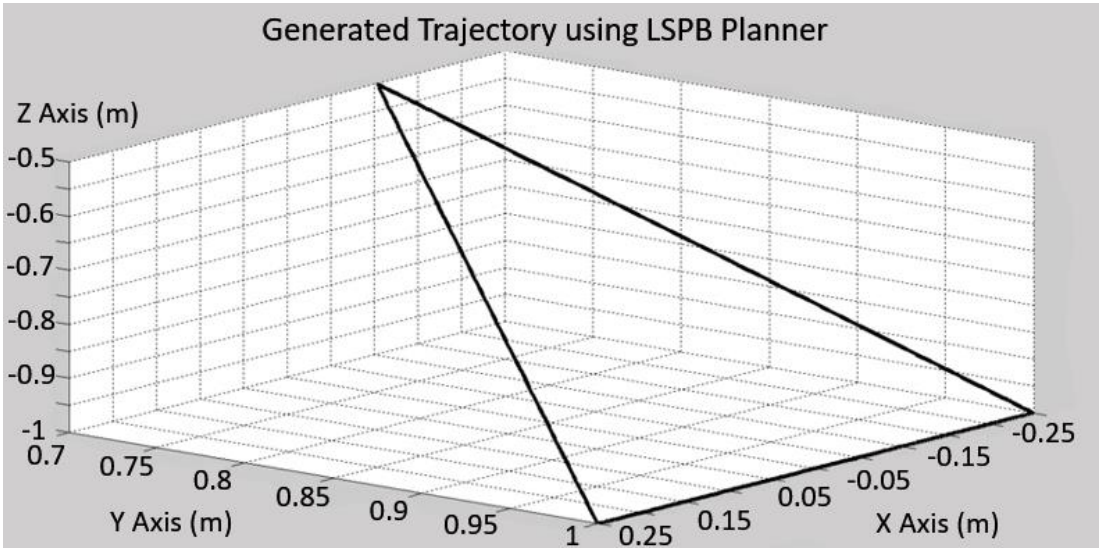


Figure 4.17 The Foot Trajectory generated using the LSPB Planner

It can be seen in Figure 4.17 that the trajectory of the complete leg cycle is linear since all the intermediate velocities between consecutive segments were set to $0\ \mathrm{ms}^{-1}$. Using heuristics, the optimum intermediate velocities may be calculated given that only initial and final velocities are $0\ \mathrm{ms}^{-1}$. Figure 4.18 shows the position, velocity, and acceleration calculated using heuristics, notice that the intermediate velocities are no longer equal to $0\ \mathrm{ms}^{-1}$. Consequently, the generated foot trajectory, shown by Figure 4.19, is an arc instead of a straight line. This result shows that the desired foot trajectory is created. MATLAB Script 8-7 (in Appendix) was used to generate Figures 4.17 and 4.19.

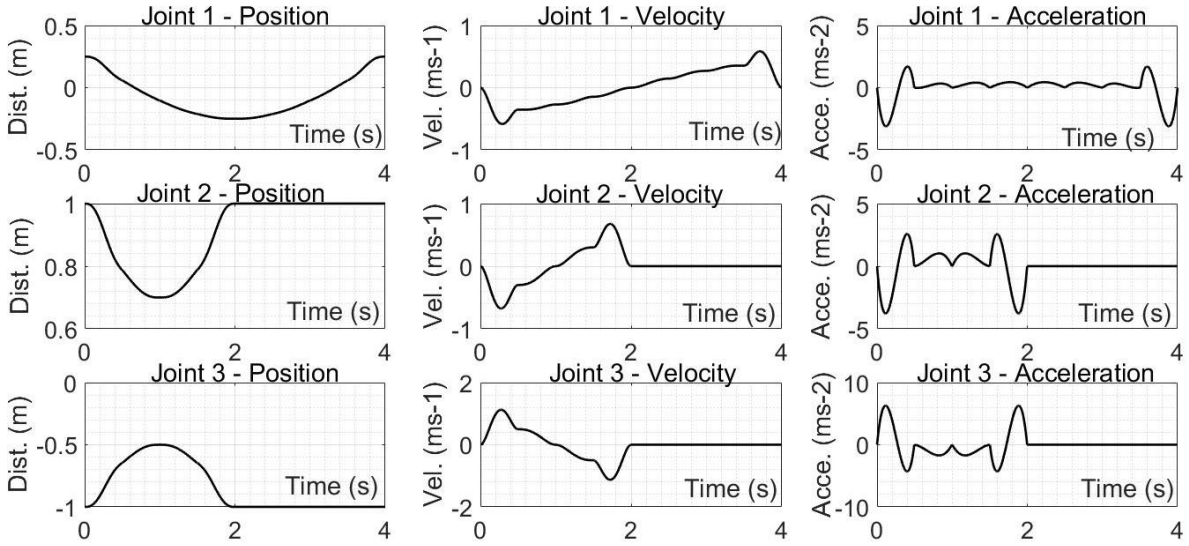


Figure 4.18 Position, velocity, and acceleration calculated using heuristics

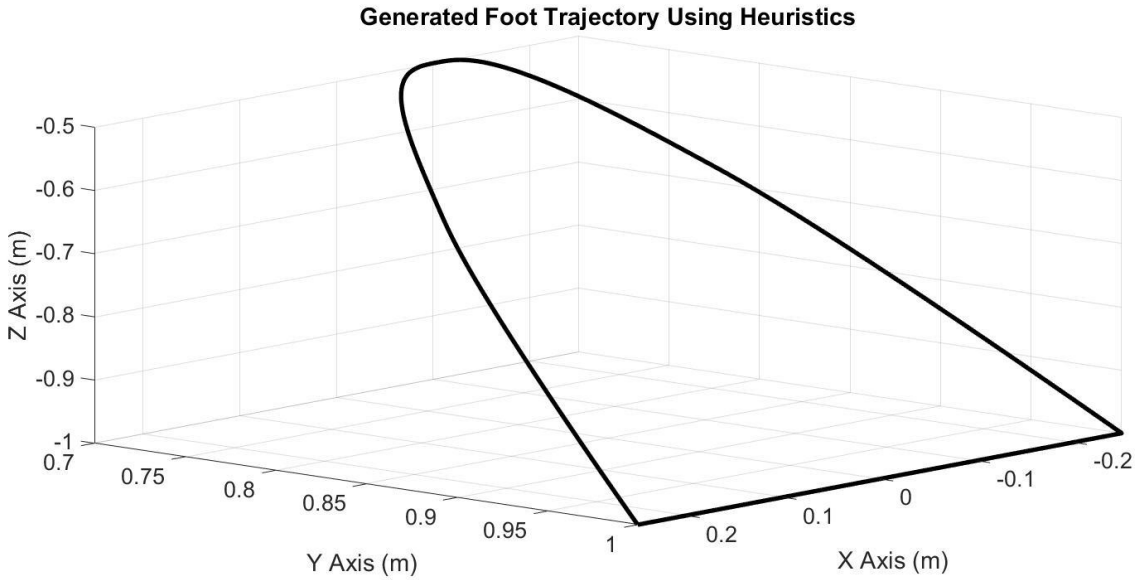


Figure 4.19 The Foot Trajectory generated using heuristics

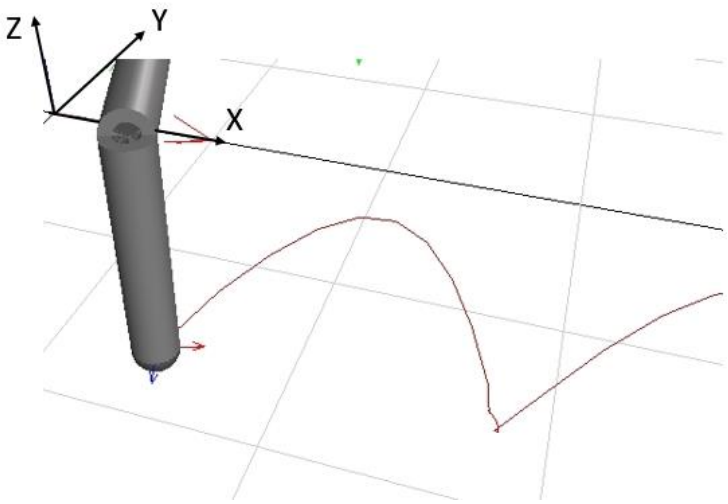


Figure 4.20 MapleSim implementation of the generated trajectory

Figure 4.20 shows the simulation of the foot trajectory which was obtained using quantic polynomial that is given by Equation 34. Notice that, during the swing phase, the leg end effector approaches the body in the direction of Y axis, shifting the mass towards the COG, reduces the mass distribution and hence minimises the angular momentum [34]. Due to the law of conservation of total momentum, the reduced angular momentum must be compensated by the speed of the motion i.e. the end effector pursues on the given trajectory at a higher velocity [34].

When the robot is navigating on an uneven terrain, a path planner algorithm needs to be implemented in order to compute the Cartesian coordinates of the end effector for a given number of intermediate segments within the trajectory. These Cartesian coordinates are then connected and a collision-free trajectory is obtained.

As it was mentioned in the literature review, artificial potential fields, probabilistic roadmap method (PRM) and rapidly exploring random trees (RRT) algorithms are widely used to generate a point-to-point trajectory [14] [16] [5]. Using artificial potential fields may be quite problematic since the existence of a local minima in the field causes the algorithm to get trapped [16]. As a result, random motion algorithms appear to be better alternatives since these algorithms are often quicker in terms of computation [16]. The PRM algorithm is more advantageous over RRT which spreads omni-directionally, mainly because relatively fewer points need to be tested in order to generate the trajectory between initial configuration, q_0, and final configuration, q_f, which reduces the computation time. Computation time needs to be taken into account since this process is repeated in every footstep and hence an algorithm that creates less computational burden was chosen [12].

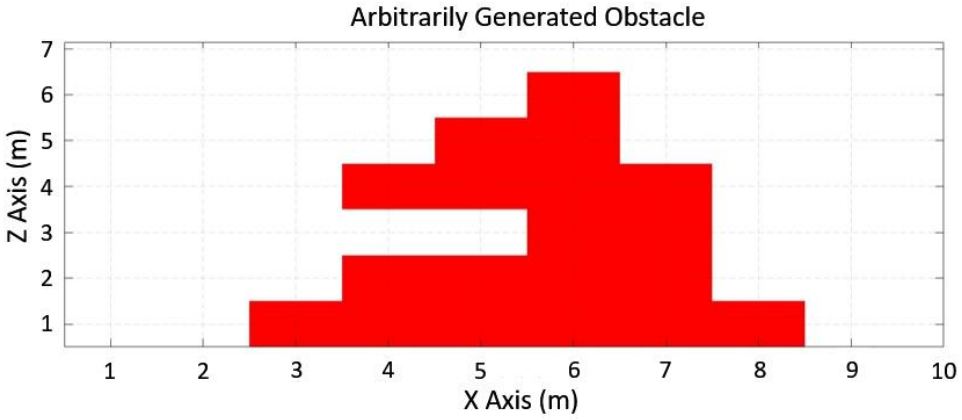


Figure 4.21 Arbitrarily generated obstacle to test the PRM algorithm

Figure 4.21 shows an obstacle that is located within the configuration space of the robot, Q. An occupancy grid matrix is generated in a similar manner to the body path

planner where the Q is discretised by representing the obstacle region, Q_{obs}, as 1 and free configuration space, Q_{free}, as 0.

Equation 41 illustrates the occupancy grid of the obstacle shown in Figure 4.21 using a 10 by 10 matrix, the matrix dimension may be extended to 100 by 100 to improve the accuracy however the computational burden on the processor would be increased as a trade-off. Notice that the matrix given by Equation 41 is vertically inverted since the 1st row and 1st column corresponds to the origin $(0,0)$ of the Cartesian coordinate system.

$$Map = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (41)$$

The chosen path planner algorithm, PRM, generates a set of random configurations within Q then the end effector configurations that lie within Q_{obs} are neglected and the configurations that are located within the borders of the Q_{free} are connected by straight lines to form pairs of configurations given that the lines do not violate the Q_{obs} [16]. The lines that introduce a collision, i.e. violate the Q_{obs} are also neglected. Then, the Cartesian coordinates of the configurations that are located on the shortest roadmap connecting the q_0 and q_f are used by the trajectory planner to generate the collision free foot trajectory that lands on the position projected by the footstep planner.

Figure 4.22 shows the set of configurations that needs to be followed to prevent a collision with the arbitrarily generated obstacle, where the randomly chosen configurations are denoted by blue nodes and the configurations that lie on the shortest roadmap are denoted by green nodes. In this particular example, the q_0 and q_f are chosen as $(2,1)$ and $(9,1)$, respectively, whereas in the simulation the initial and final configurations are provided by the footstep planner.

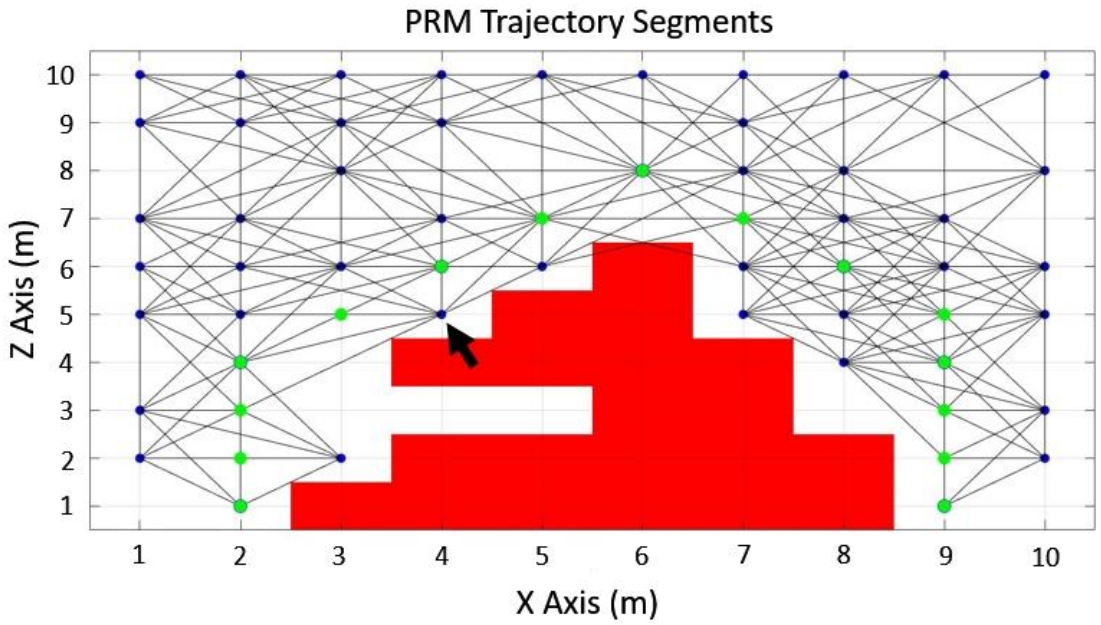


Figure 4.22 The intermediate end effector positions that connect q_0 and q_f.

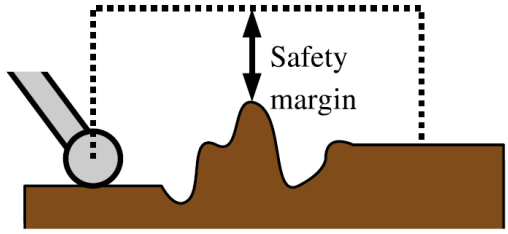


Figure 4.23 [8] Safety margin

The PRM planner needs to be specified with a safety margin, i.e. minimum distance that the green nodes can be positioned. An example is shown in Figure 4.23. In practice, a zone surrounding the obstacle is formed which the planner treats as an expanded Q_{obs} and hence it will not be violated. An example of the expanded Q_{obs} is provided in Figure 4.24. To illustrate, the node that is located on $(4,5)$ in Figure 4.22 was not chosen by the planner since its location lies within the specified safety margin of 0.3m.

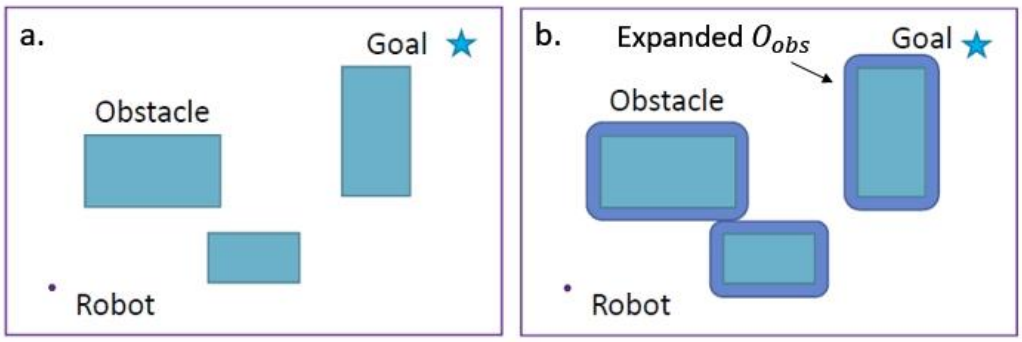


Figure 4.24 The obstacle that is expanded by a safety margin, adapted from [31]

Note that since PRM is a random procedure each iteration of the code which is given in MATLAB Script 8-5, may give a different set of configurations for the same obstacle, as shown in Figure 8.3 (in Appendix).

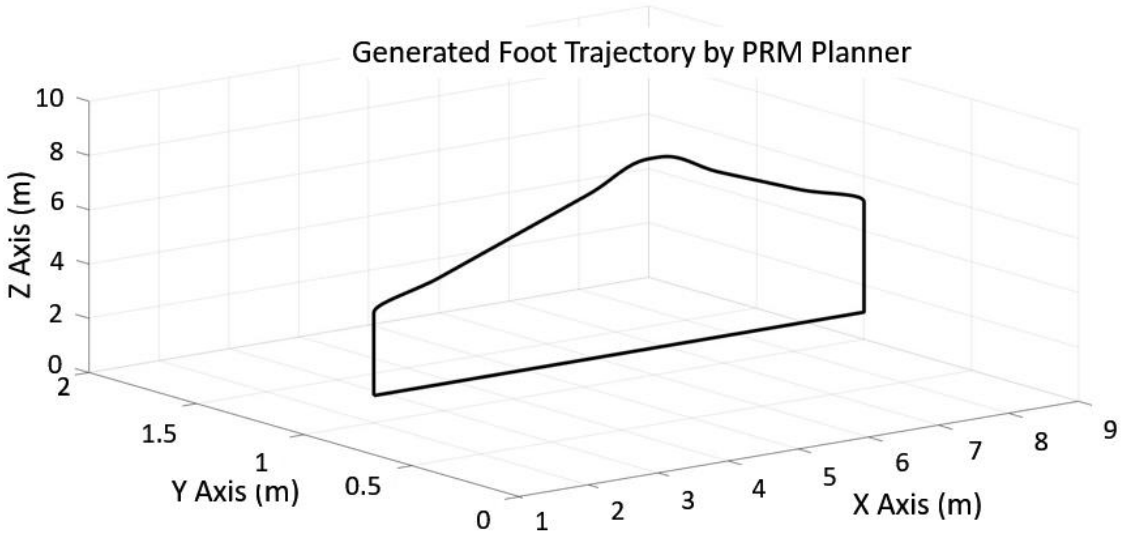


Figure 4.25 The complete foot trajectory

Figure 4.25 shows the generated foot trajectory by using the intermediate leg configurations that were given by the PRM Planner. This trajectory is used to compute the 3 joint angles, θ_1, θ_2, and θ_3 by using the inverse kinematics equations to make the leg step over the obstacle and land to the projected q_f.

Figure 4.26 shows the simulation result that was generated in the MapleSim environment. The black line shows the path trace of the motion and it can be seen that the leg steps over the obstacle as expected. The same method is used by the hexapod robot to compute the trajectory over an obstacle while navigating on an uneven terrain.

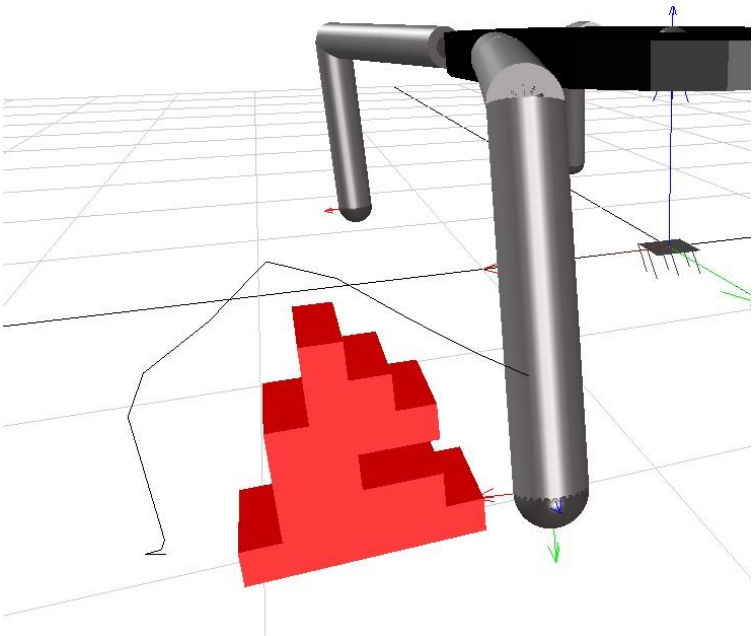


Figure 4.26 MapleSim implementation of the generated foot trajectory

4.2.2. Body Trajectory Planner

As it was stated in Section 3.2., the adapted walking gait and dynamic stability approaches are used by the MapleSim model to maintain the stability of robot when the terrain becomes steeper or the locomotion speed is increased so that the static stability may no longer be preserved. The results showed that the dynamic stability approach appears to be more advantageous over adaptive walking gait since it improves the locomotion speed of the robot while maintaining its stability.

Adaptive Walking Gait

In the MapleSim simulation, the SM of the motion, which is an indication of how close the robot is to be statically unstable, was used to alter the walking gait depending on the slope of the terrain that the robot is located on. For instance, the walking gait is switched from tripod gait to wave gait in order to increase the area of the support polygon and hence the stability of the robot.

Figure 4.27 shows the support polygon that is formed between the ground contact points during wave gait where a single leg is in the swing phase at a time. It can be seen that the wave gait is intrinsically more stable since the area of the support polygon is greater comparing to tripod gait.

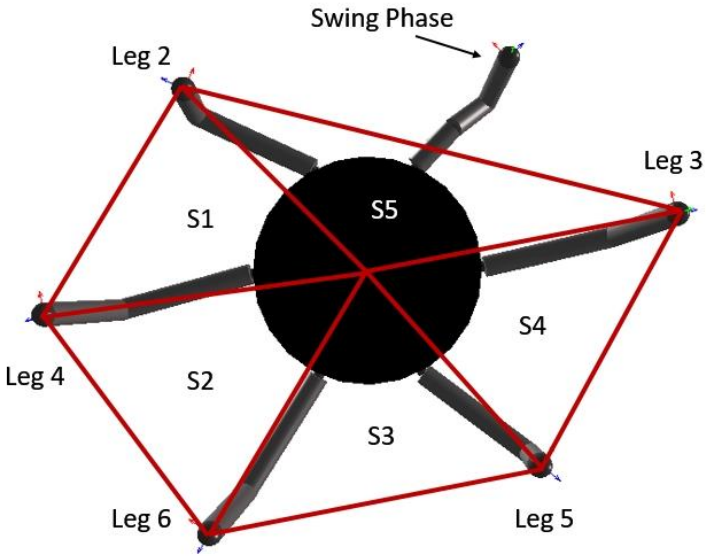


Figure 4.27: Bottom view of the Hexapod Robot during wave gait

Figure 4.28 shows the transition from tripod gait to wave gait to improve the stability. However, this transition gave rise to a trade-off between locomotion speed and stability e.g. Figure 4.28-c shows that fewer steps are taken. The MapleSim code that was used to generate the adaptive walking gait is shown in Figure 8.2 (in Appendix).

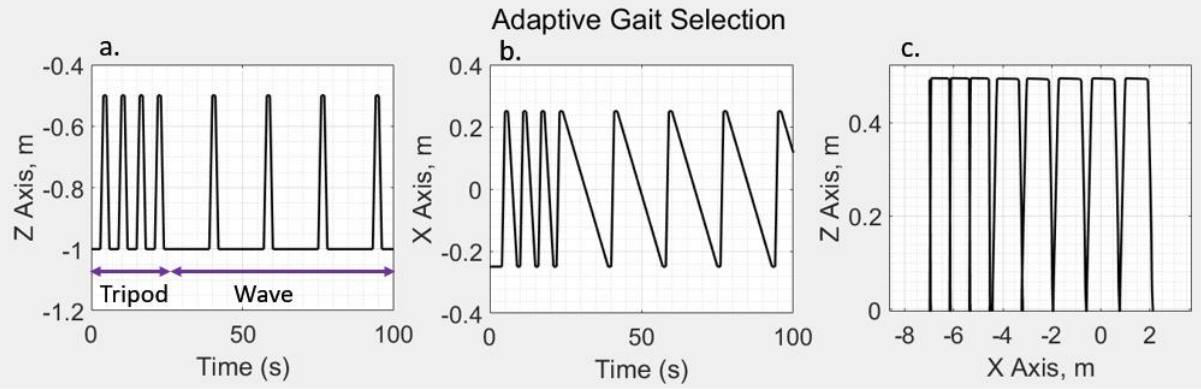


Figure 4.28 The transition from tripod gait to wave gait.

Dynamic Stability

The static stability criterion that was covered in Section 3.2 does not take the effect of external forces into account which are caused by the inertial acceleration of the robot. For a legged robot that is navigating on an uneven terrain, the vertical projection of the COG is expected to be located outside or on the border of the support polygon [20].

Therefore, dynamic stability criterion needs to be implemented which is based on computing the optimum acceleration profile of the COG that keeps the zero moment point (ZMP) within the borders of the support polygon during a complete gait cycle [35] [18]. In other words, even though the vertical projection of the COG is located outside the support polygon, the robot is balanced by the external forces that are created during motion. ZMP is defined by [36] as "the point on the ground surface about which the horizontal component of the moment of ground reaction force is zero." The positions of ZMP and COG coincide with each other when the robot is stationary where COG is assumed to be the geometric centre of the robot [35] [31].

The body trajectory planner, say in tripod gait, takes the estimated positions of the next 3 footholds which were calculated by the foot step planner and computes a COG trajectory given that ZMP of the robot is kept within the support polygon [18]. The relationships between the position of the ZMP along the X axis, x_{ZMP} and the Y axis, y_{ZMP} are given by Equations 42 and 43, respectively.

$$x_{ZMP} = x_m - \frac{z_m \ddot{x}_m}{\ddot{z}_m + \mathrm{g}} \qquad (42)$$

$$y_{ZMP} = y_m - \frac{z_m \ddot{y}_m}{\ddot{z}_m + \mathrm{g}} \qquad (43)$$

where x_m, y_m and z_m are the X, Y and Z coordinates of the COG and g is the gravitational acceleration [18].

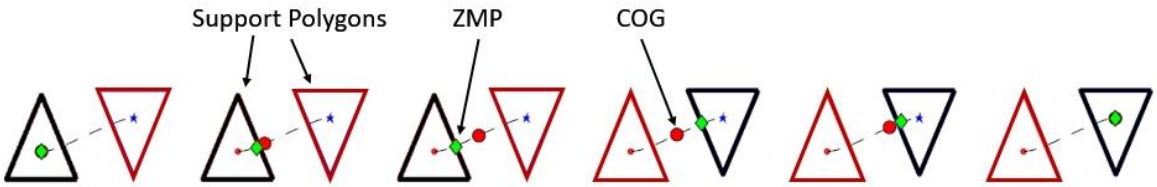


Figure 4.29 The trajectory of COG and ZMP during tripod gait, adapted from [18]

Figure 4.29 shows the trajectory that the COG traverses from previous support polygon to next support polygon while ZMP is kept within the current support polygon which is shown by the black triangle [18].

4.3. Low Level Controller

In real life, the planned footstep placement and foot trajectories may not be executed as anticipated due to a number of affects such as friction, jamming, slippage, and meshing. Consequently, the stability of the robot is jeopardized and hence the robot locomotion is no longer accurate. In order to overcome the mentioned problems, 2 low level controllers are implemented [18].

4.3.1. Inverse Dynamics

Inverse dynamics is a nonlinear control technique that provides a better trajectory tracking by calculating the required joint actuator torques that need to be generated by the motors to achieve a given trajectory [25] [16].

As it is given in Section 3.4.1, the matrix form of the dynamics of a robot leg is given by Equation 44 where the F and J terms are neglected [25].

$$M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + G(q) = \tau \tag{44}$$

For a given q_0 and v_0 the nonlinearity of the system can be vanished, making the system linear and hence the exact amount of torque that needs to be generated by the joint actuator to overcome the inertia of the actuator can be calculated. If the q_0 and v_0 match with the desired q_0 and v_0 then the projected trajectory will be tracked as expected [37].

Equation 45 shows the control law expression where a^q is a new input to the system. The numeric value of a^q is calculated by Equation 46 where q_d is the desired trajectory, q is the actual trajectory, and K_p and K_d are position and velocity gains, respectively [16].

$$\tau = M(q)a^q + C(q,\dot{q})\dot{q} + G(q) \quad (45)$$

$$a^q = \ddot{q}_d + (\dot{q}_d - \dot{q})K_d + (q_d - q)K_p \quad (46)$$

Since the mass matrix, M, is invertible, equating Equation 45 to Equation 46 gives the expression in Equation 47 which shows that the new system is linear as it is given by Equation 48.

$$M(q)\ddot{q}_d = M(q)a^q \quad (47)$$

$$\ddot{q}_d = a^q \quad (48)$$

Figure 4.30 demonstrates the inner loop and outer loop control architecture where the output of the system yields the joint space signal which enables a better trajectory tracking given that the feedback gains K_p and K_d are tuned as appropriate.

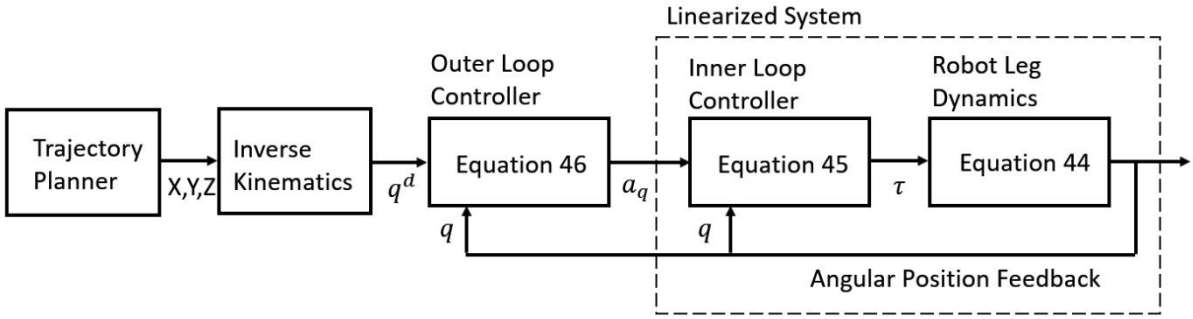


Figure 4.30: Inner Loop – Outer Loop Linearized Control, adapted from [16]

4.3.2. Stability Detection and Recovery

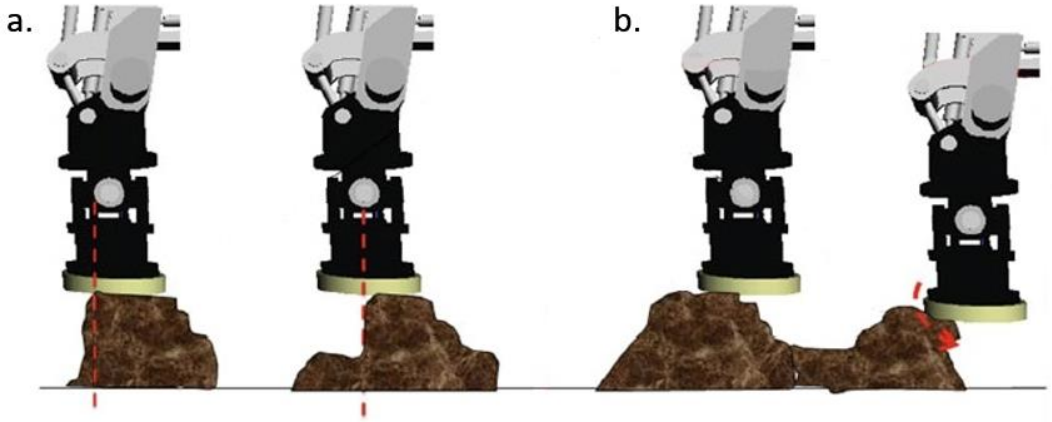


Figure 4.31: The foot positions that are likely to cause a slippage, adapted from [2]

Figure 4.31 shows the foothold positions that the robot leg may slip during the stance phase. Figure 4.31-b has a higher chance of slippage than the foot positions that are shown in Figure 4.31-a. For such cases, a recovery controller needs to be implemented for real life applications where conditions are not ideal, this controller is

particularly suitable for the cases in which the robot slips or an unexpected obstacle happens to be present within the workspace of the robot leg. The recovery controller provides a reflex based reaction to the 3 cases that are shown in Figure 4.33.

- In Figure 4.32-a, the leg slips from position 1 to position 2 then the recovery controller takes action and leg is moved to position 3 to minimise the slippage.
- In Figure 4.32-b, the leg encounters an unexpected obstacle at position 2 and leg is raised higher to overcome the obstacle and positioned to position 3.
- In Figure 4.32-c, no contact is detected at the expected location 2 and the leg seeks for another foothold and eventually detects position 3.

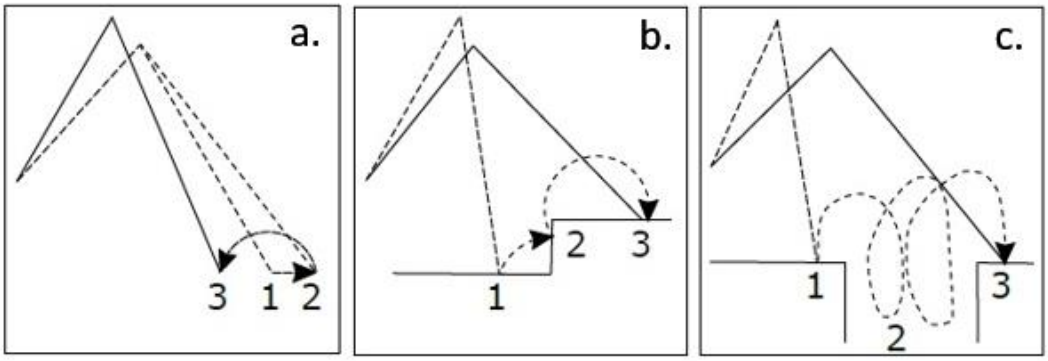


Figure 4.32: Recovery reflexes, adapted from [4]

5. Simulation Results

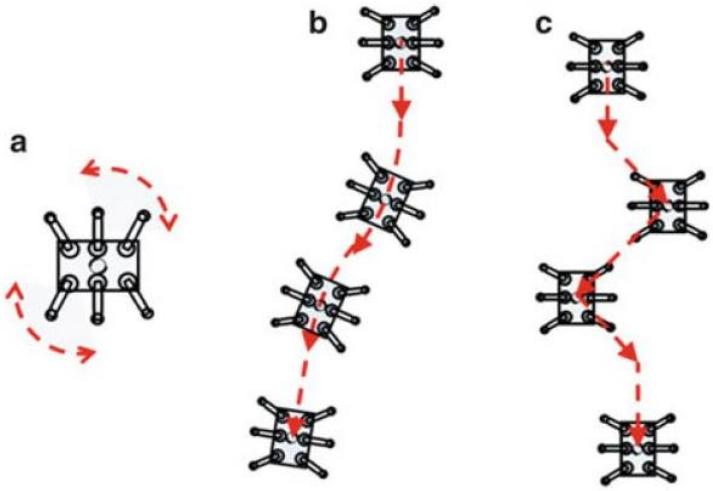


Figure 5.1 Omni-directional Locomotion Types, adapted from [2]

Figure 5.1 shows the 3 omnidirectional locomotion types that can be implemented in hexapod robots to change direction while navigating within an environment. MapleSim simulation environment was used for developing robot models for each omnidirectional locomotion type and for testing the hierarchical control architecture that has been presented in Chapter 4.

5.1. Centroid Rotation

Figure 5.1-a shows a hexapod robot rotating about its centre, such manoeuvrability provides a great advantage to legged robots when changing direction within a limited movement area [2]. Figure 5.2 shows the simulation results that have been generated by implementing the centroid rotation. It can be seen that hexapod robot is capable of rotating both clockwise and anticlockwise directions and it is possible to increase the step size which speeds up the process but reduces the precision. Notice that in Figure 5.2, the step size of the bottom hexapod is greater than the top hexapod.

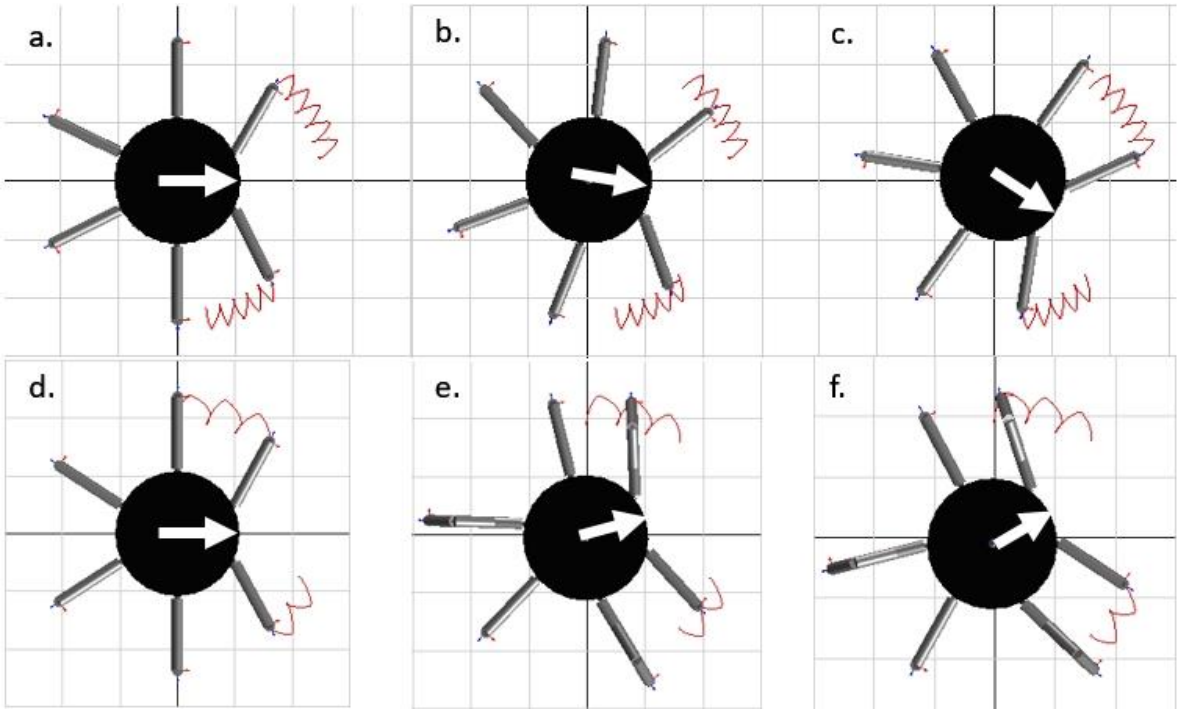


Figure 5.2 MapleSim implementation of centroid rotation.

5.2. Slalom

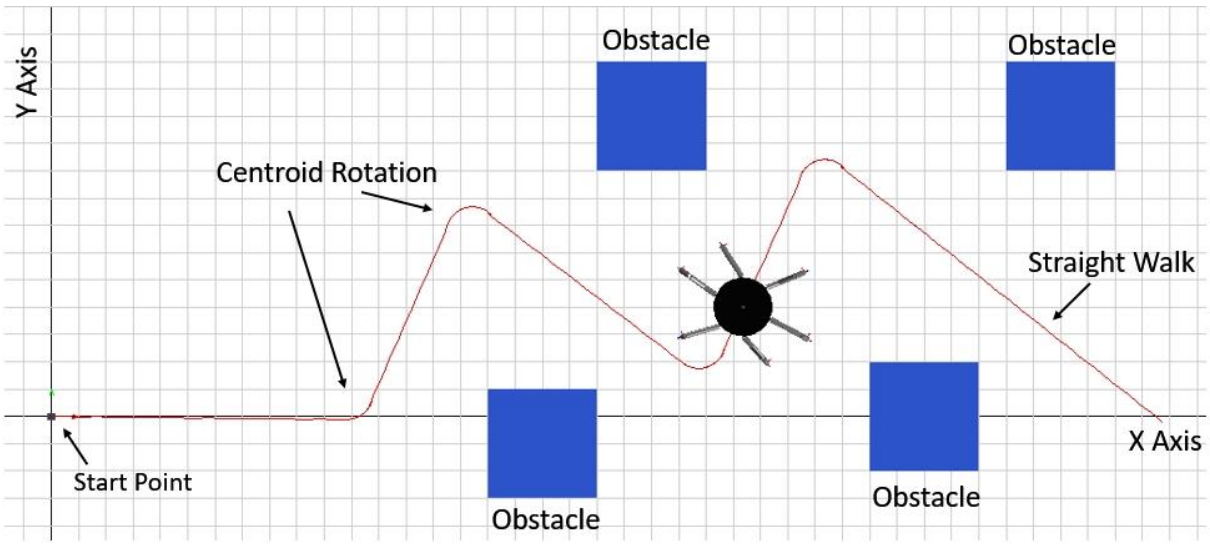


Figure 5.3 MapleSim implementation of slalom locomotion.

Figure 5.1-b shows the slalom movement of the hexapod robots. The robot plans its motion as a combination of centroid rotation and straight walking. It partitions its body path into linear lines and rotates about its centre between each linear line segment to align itself with the next linear region. Figure 5.3 shows the slalom implementation in the MapleSim environment where the red line shows the path trace that the robot has followed. The code that was used to deduce the direction of rotation is provided in Figure 8.4 (in Appendix).

5.3. Pedestrian-lane-change

Figure 5.1-c shows that the robot leg is moved diagonally depending on the direction of motion and the body maintains its initial orientation at all times. This particular locomotion type is named as pedestrian-lane-change by [2] and the results obtained from the MapleSim implementation are provided in Figures 5.4-5.6.

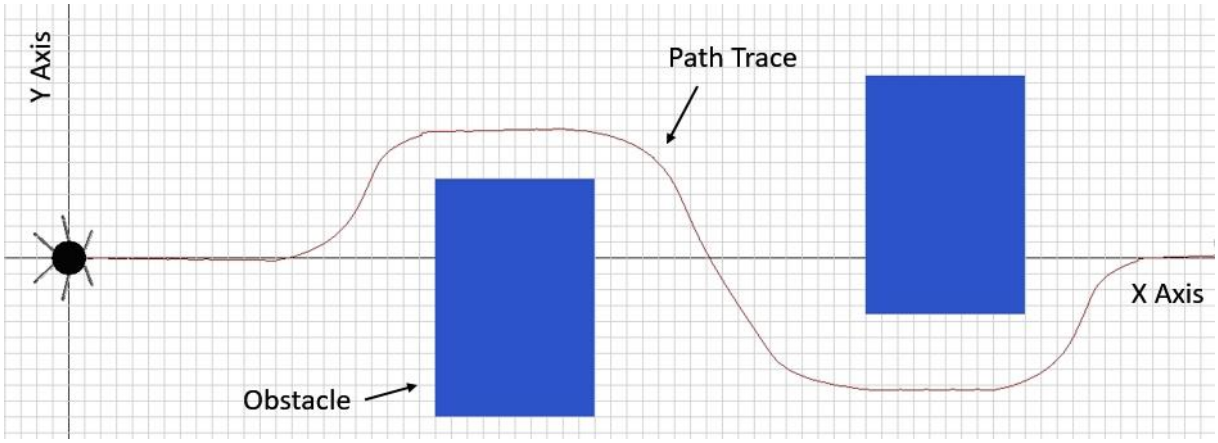


Figure 5.4 Top view of the pedestrian-lane-change locomotion.

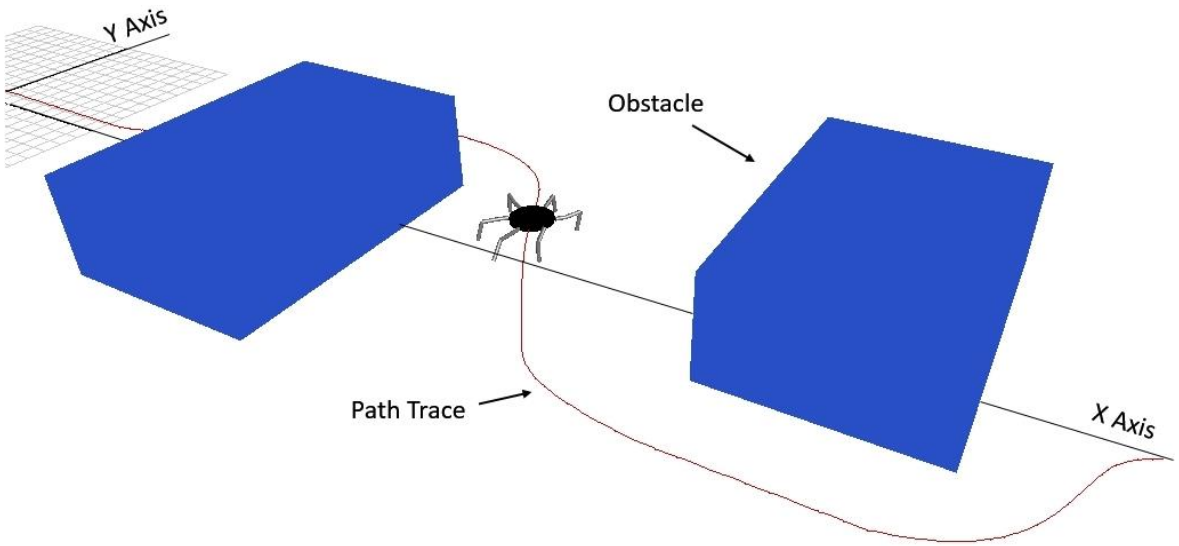


Figure 5.5 3D view of the pedestrian-lane-change locomotion.

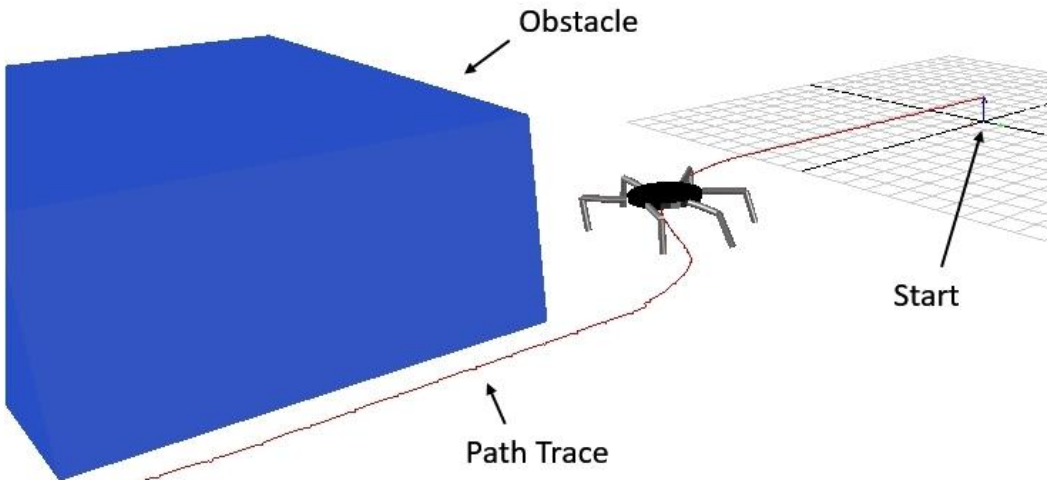


Figure 5.6 3D view of the pedestrian-lane-change locomotion.

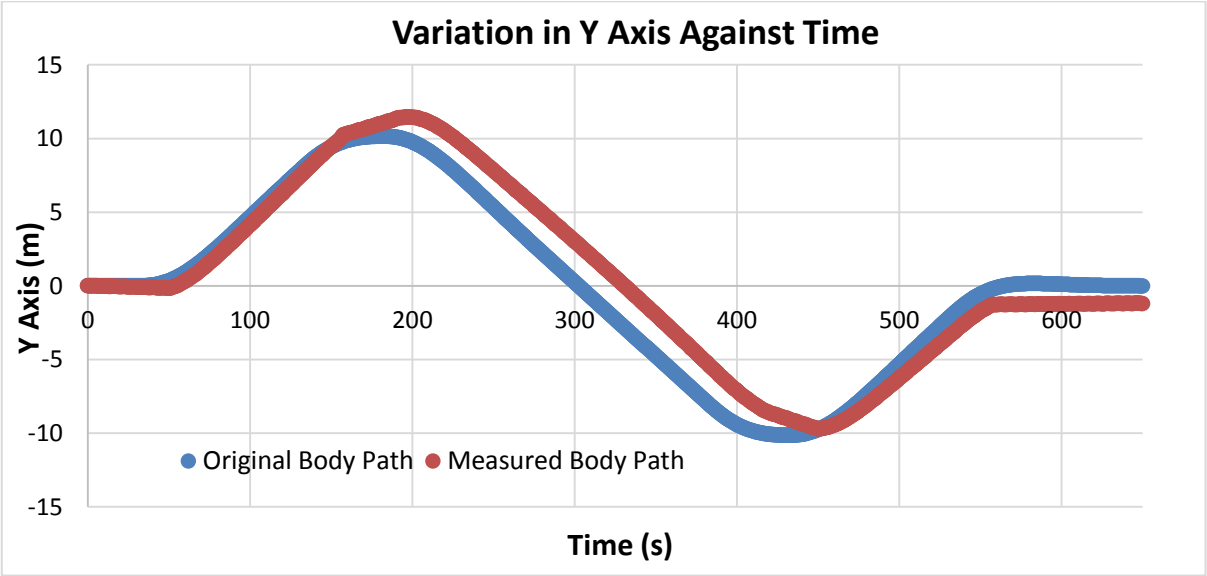


Figure 5.7 The comparison of the variation in Y axis against time.

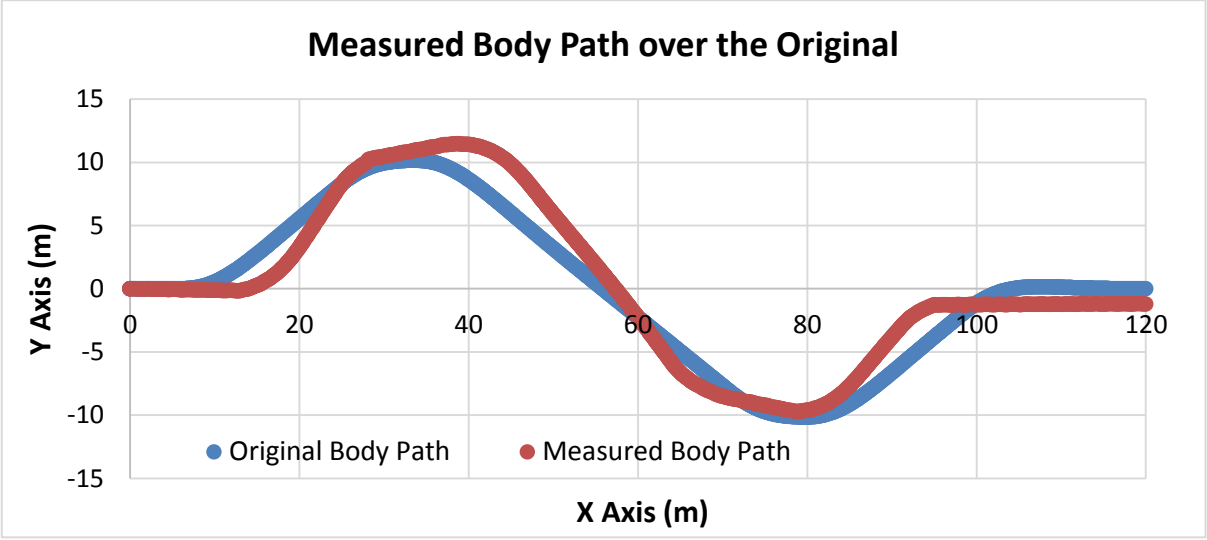


Figure 5.8 The comparison of the variation in Y axis against X axis.

Figures 5.7 and 5.8 show that the body path that the robot has followed. Considering that the wingspan of the robot is about 6 meters, the measured body path of the robot, shown in red, is almost the same as the actual body path that was calculated by body path planner, shown in blue.

6. Further Work Required

In future, a biologically inspired self-configuration algorithm may be implemented for the cases where the robot losses one or more of its legs. Swarm Intelligence based robot reconfiguration (SIRR) method is delivered in [17]. An example is given in Figure 6.1 where hexapod reconfigures itself as a quadruped after losing Legs 1 and 5.

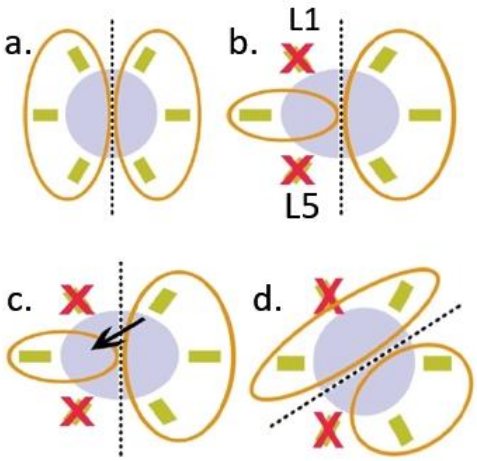


Figure 6.1 Robot Reconfiguration, adapted from [17]

7. Conclusion

This is a simulation-based project which aimed to model a hexapod robot in the MapleSim environment and to implement different path planning algorithms and a hierarchical control architecture. This report has presented the theoretical background, design, implementation, and control of a circular body shaped hexapod robot. The simulation model has successfully navigated between a start point and a given target point on a level terrain with randomly placed obstacles. It has been discovered and confirmed with a Maplesoft employee that the Contact Library in MapleSim is not suitable for legged-locomotion on an uneven terrain whereas wheeled-locomotion on an uneven terrain can be implemented by using the Tire Library. For that reason, an uneven terrain was constructed in the MATLAB environment and it was shown that the Footstep planner has successfully worked on an uneven terrain as well as a level terrain. Since the rest of the control hierarchy remains the same, it can be said that the provided system will potentially work on an uneven terrain.

8. Appendix

8.1. Additional Visuals

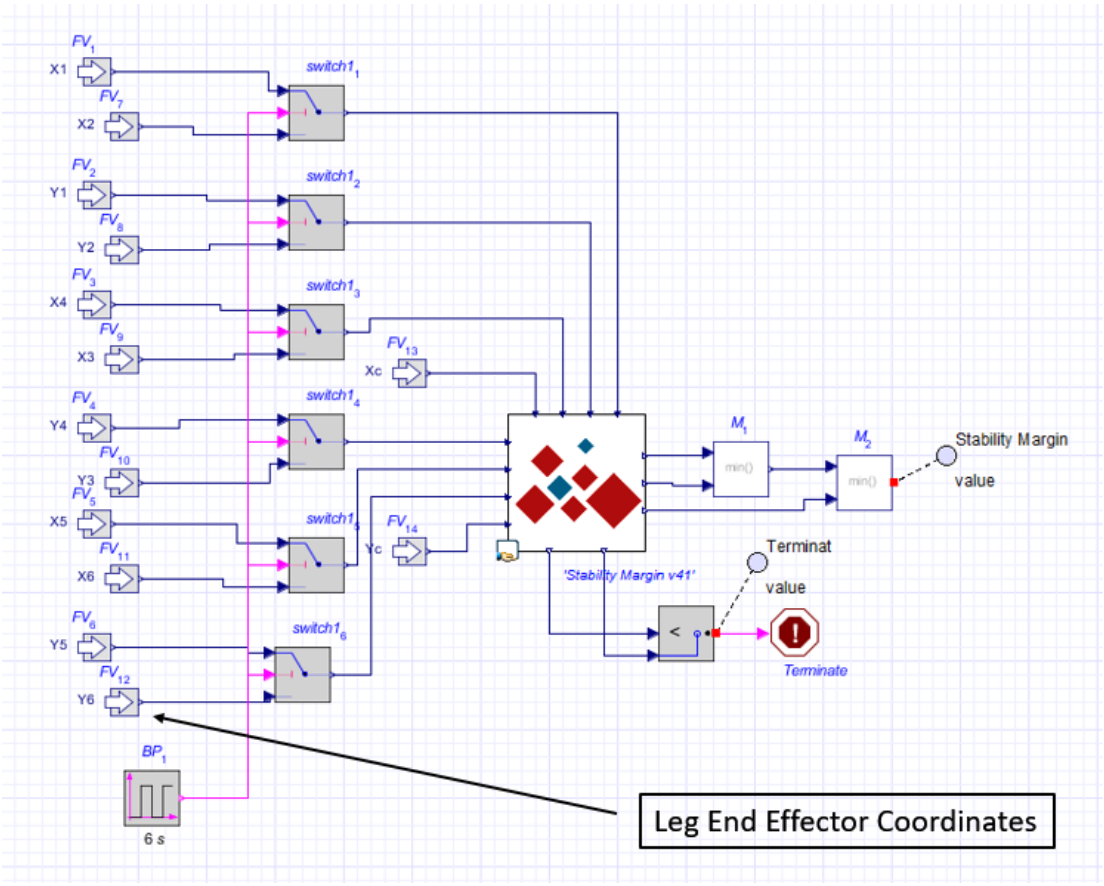


Figure 8.1 The MapleSim code that was used for calculating the SM

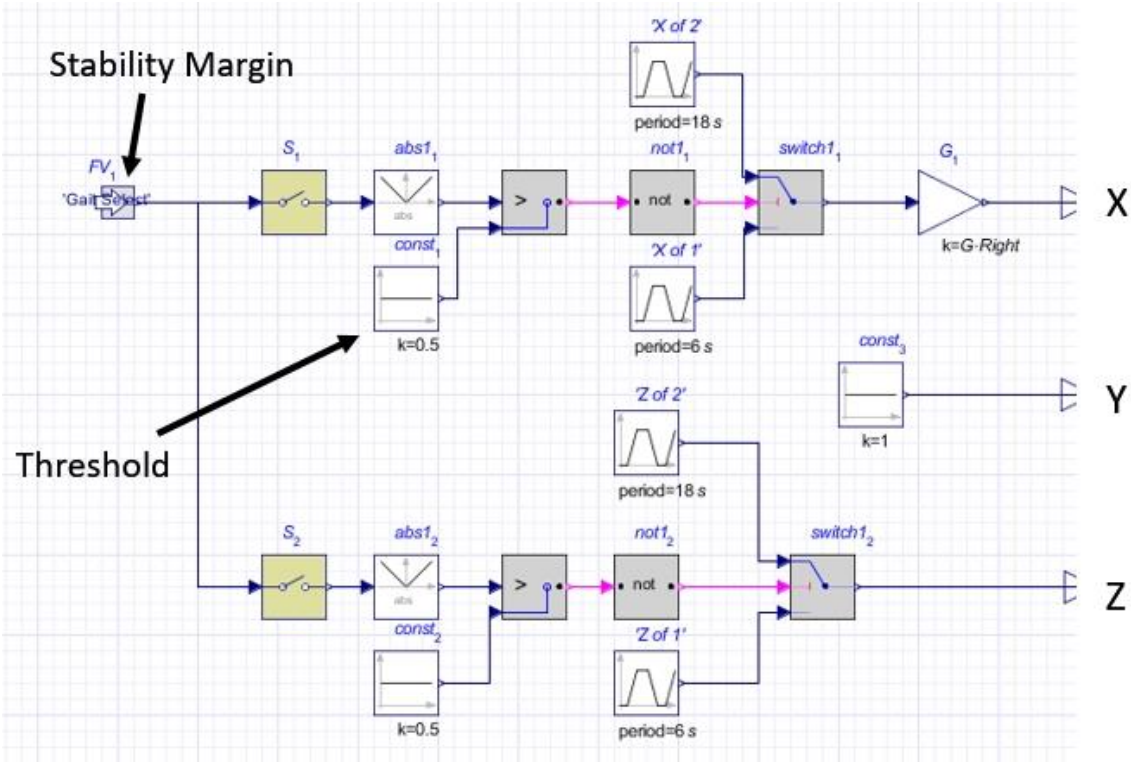


Figure 8.2 The MapleSim code that was used for adaptive gait selector

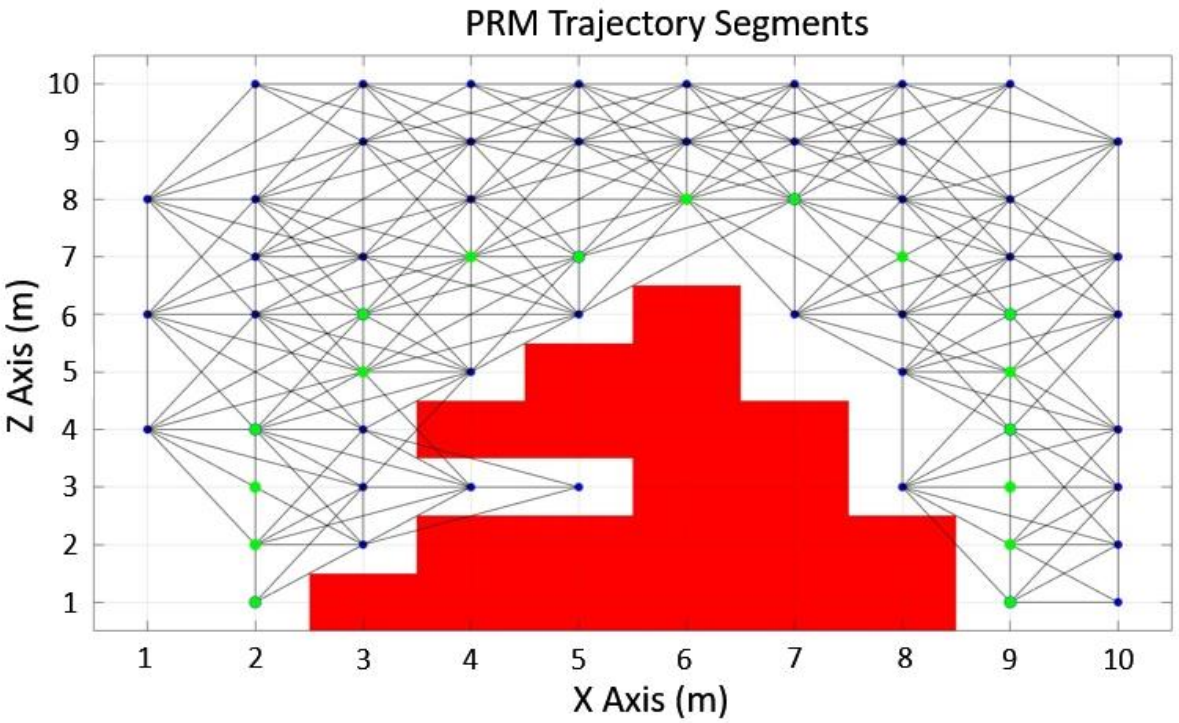


Figure 8.3 The new set of intermediate postions that are obtained in the second iteration of the PRM Planner

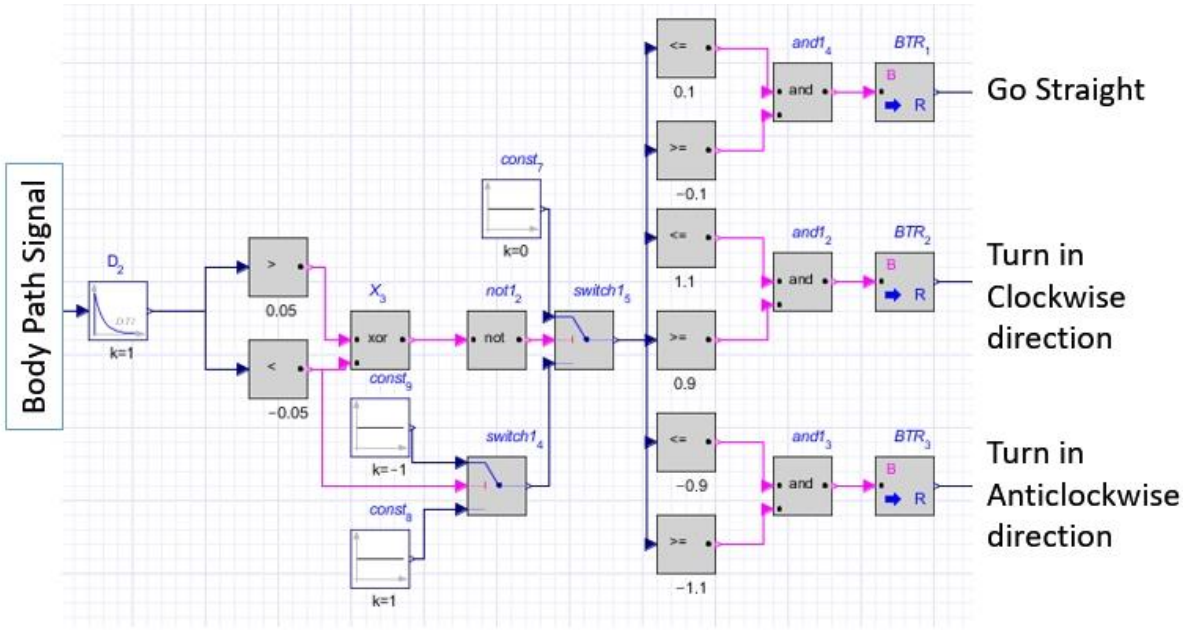


Figure 8.4 The code that was used for direction of rotation detection

8.2. Additional Equations

$$J_1 = \begin{bmatrix} \frac{-1}{2} L_1 \cos(q_1) & 0 & 0 \\ \frac{-1}{2} L_1 \sin(q_1) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} -L_1 \cos(q_1) - \frac{1}{2} L_2 \cos(q_2) \cos(q_1) & \frac{1}{2} L_2 \sin(q_2) \sin(q_1) & 0 \\ -L_1 \sin(q_1) - \frac{1}{2} L_2 \cos(q_2) \sin(q_1) & -\frac{1}{2} L_2 \sin(q_2) \cos(q_1) & 0 \\ 0 & \frac{1}{2} L_2 \cos(q_2) & 0 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} -L_1 \mathrm{c}(q_1) - L_2 \mathrm{c}(q_2)\,\mathrm{c}(q_1) - \frac{1}{2} L_3 \mathrm{c}(q_2 + q_3)\,\mathrm{c}(q_1) & L_2 \mathrm{s}(q_2)\,\mathrm{s}(q_1) + \frac{1}{2} L_3 \mathrm{s}(q_2 + q_3)\,\mathrm{s}(q_1) & \frac{1}{2} L_3 \mathrm{s}(q_2 + q_3)\,\mathrm{s}(q_1) \\ -L_1 \mathrm{s}(q_1) - L_2 \mathrm{c}(q_2)\,\mathrm{s}(q_1) - \frac{1}{2} L_3 \mathrm{c}(q_2 + q_3)\,\mathrm{s}(q_1) & -L_2 \mathrm{s}(q_2)\,\mathrm{s}(q_1) - \frac{1}{2} L_3 \mathrm{s}(q_2 + q_3)\,\mathrm{c}(q_1) & -\frac{1}{2} L_3 \mathrm{s}(q_2 + q_3)\,\mathrm{c}(q_1) \\ 0 & L_2 \mathrm{c}(q_2) + \frac{1}{2} L_3 \mathrm{c}(q_2 + q_3) & \frac{1}{2} L_3 \mathrm{c}(q_2 + q_3) \end{bmatrix}$$

where L_1, L_2, and L_3 are the lengths of links 1, 2, and 3 respectively.

8.3. MATLAB Code

```
%forward kinematics

T=[0,0,0];   %t[t1,t2,t3]
L=[0.5,0.75,1];  %l=[Link1,Link2,Link3]

%DH Table Parameters
theta_1 = (T(1)+90)*pi/180; d1 = 0; a1 = L(1); alpha_1 = pi/2;
theta_2 = T(2)*pi/180; d2 = 0; a2 = L(2); alpha_2 = 0;
theta_3 = (T(3)-90)*pi/180; d3 = 0; a3 = L(3); alpha_3 = -pi/2;
theta_4 = -pi/2; d4 = 0; a4 = 0; alpha_4 = pi/2;

%Homogeneous transformation matrices
H01=[cos(theta_1)   -sin(theta_1)*cos(alpha_1)  sin(theta_1)*sin(alpha_1)   a1*cos(theta_1);
     sin(theta_1)   cos(theta_1)*cos(alpha_1)   -cos(theta_1)*sin(alpha_1)  a1*sin(theta_1);
     0              sin(alpha_1)                cos(alpha_1)                d1;
     0                     0                        0                     1];

H12=[cos(theta_2)   -sin(theta_2)*cos(alpha_2)  sin(theta_2)*sin(alpha_2)   a2*cos(theta_2);
     sin(theta_2)   cos(theta_2)*cos(alpha_2)   -cos(theta_2)*sin(alpha_2)  a2*sin(theta_2);
     0              sin(alpha_2)                cos(alpha_2)                d2;
     0                     0                        0                     1];

H23=[cos(theta_3)   -sin(theta_3)*cos(alpha_3)  sin(theta_3)*sin(alpha_3)   a3*cos(theta_3);
     sin(theta_3)   cos(theta_3)*cos(alpha_3)   -cos(theta_3)*sin(alpha_3)  a3*sin(theta_3);
     0              sin(alpha_3)                cos(alpha_3)                d3;
     0                     0                        0                     1];

H34=[cos(theta_4)   -sin(theta_4)*cos(alpha_4)  sin(theta_4)*sin(alpha_4)   a4*cos(theta_4);
     sin(theta_4)   cos(theta_4)*cos(alpha_4)   -cos(theta_4)*sin(alpha_4)  a4*sin(theta_4);
     0              sin(alpha_4)                cos(alpha_4)                d4;
     0                     0                        0                     1];

end_effector_frame=[0;0;0;1];
HT=H01*H12*H23*H34*end_effector_frame;
```

MATLAB Script 8-1

```
%entire work space

l=[0.5,0.75,1];  %link lengths
M=[];

for a=-90:10:90
    for b=-90:15:90
        for c=-60:15:135
            T1= a; T2= b; T3= c;
            %DH Table Parameters
            theta_1 = (T1+90)*pi/180; d1 = 0; a1 = l(1); alpha_1 = pi/2;
            theta_2 = T2*pi/180; d2 = 0; a2 = l(2); alpha_2 = 0;
            theta_3 = (T3-90)*pi/180; d3 = 0; a3 = l(3); alpha_3 = -pi/2;
            theta_4 = -pi/2; d4 = 0; a4 = 0; alpha_4 = pi/2;

H01=[cos(theta_1),-sin(theta_1)*cos(alpha_1),sin(theta_1)*sin(alpha_1),a1*cos(theta_1);
     sin(theta_1),cos(theta_1)*cos(alpha_1),-cos(theta_1)*sin(alpha_1),a1*sin(theta_1);
     0,sin(alpha_1),cos(alpha_1),d1;
     0,0,0,1];

H12=[cos(theta_2),-sin(theta_2)*cos(alpha_2),sin(theta_2)*sin(alpha_2),a2*cos(theta_2);
     sin(theta_2),cos(theta_2)*cos(alpha_2),-cos(theta_2)*sin(alpha_2),a2*sin(theta_2);
     0,sin(alpha_2),cos(alpha_2),d2;
     0,0,0,1];

H23=[cos(theta_3),-sin(theta_3)*cos(alpha_3),sin(theta_3)*sin(alpha_3),a3*cos(theta_3);
     sin(theta_3),cos(theta_3)*cos(alpha_3),-cos(theta_3)*sin(alpha_3),a3*sin(theta_3);
     0,sin(alpha_3),cos(alpha_3),d3;
     0,0,0,1];

H34=[cos(theta_4),-sin(theta_4)*cos(alpha_4),sin(theta_4)*sin(alpha_4),a4*cos(theta_4);
     sin(theta_4),cos(theta_4)*cos(alpha_4),-cos(theta_4)*sin(alpha_4),a4*sin(theta_4);
     0,sin(alpha_4),cos(alpha_4),d4;
     0,0,0,1];

            end_effector_frame=[0;0;0;1];
            HT=H01*H12*H23*H34*end_effector_frame;
            M=[M,HT];

        end
    end
end
```

MATLAB Script 8-2

```
%% A* Path Finding
clear all
clc

% Generate Map - 0 = unocuppied, 1 = obstacle, 2 = start, 3 = goal
M = [0 0 0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 1 0 0 0 0 0;
     0 0 0 0 0 0 0 0 1 0 0 0 0 0;
     0 0 2 0 0 1 0 0 1 0 0 3 0 0;
     0 0 0 0 0 1 0 0 0 0 0 0 0 0;
     0 0 0 0 0 1 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0 0 0 0 0];

%Set the adjacency type - 1 = von Neumann, 2 = Moore
% adjac = 1;
adjac = 2;

%Call the A* function
[len,path] = A_star(M,adjac);
```

MATLAB Script 8-3, adapted from [31]

```
%% Uneven Terrain Generation and Foothold Computations
clc
clear all
M=[];
%% Symbolic Variable declaration
syms l1 l2 l3
syms q1 q2 q3 qd1 qd2 qd3
syms qx qy qz tx ty tz
syms r11 r12 r13 r21 r22 r23 r31 r32 r33 px py pz
q1c = 0;
%% FK
P03 = [ cos(q1) * (l3*cos(q2+q3) + cos(q2) * l2 + l1);...
        sin(q1) * (l3*cos(q2+q3) + cos(q2) * l2 + l1);...
        l3*sin(q2+q3) + sin(q2) * l2];
Psym = [r11 r12 r13 px; r21 r22 r23 py; r31 r32 r33 pz;0 0 0 1];
aq1 = q1c; aq2 = 0.6357; aq3 = -1.15; al1 = 0.5;al2=0.75;al3=1.0;
cp = double(subs(P03, [q1, q2, q3, l1, l2, l3], [aq1, aq2, aq3, al1, al2, al3]));

for i=0.25:0.01:2       %My Y
        for j= 0:0.1:2  %My X
            cp(1) = i;
            cp(2) = j;

            % Generation of uneven terrain
            IMIN =(abs(50*sin(2*pi*0.1*i*2))-0.25);
            IMAX =(abs(50*sin(2*pi*0.1*i*2))-0.25)+5; %Change 5 to alter variation range
            IMIN = int8(IMIN); IMAX = int8(IMAX);
            Z1=randi([IMIN,IMAX]);

            cp(3)=Z1*0.01-1;
%% IK - Algebra solution
    try
        x = cp(1);  y = cp(2);  z = cp(3);
        q1c = atan2(y, x);

        % Adding the square of element (1,4) & (2,4) in inv(H01)*Psym = T12*T23
        c3 = ( (x*cos(q1c) + y*sin(q1c) - al1)^2 + z^2 -al3^2-al2^2 )/(2*al2*al3);
        s3 = sqrt(1-c3^2);
        q3c = [atan2(s3,c3), atan2(-s3,c3)];

        if (q3c(1) > 0)
            q3n = q3c(1);
        else
            q3n = q3c(2);
        end

        % Dividing the element (1,4) & (2,4) in inv(H01)*Psym = T12*T23
        xp = x*cos(q1c) + sin(q1c)*y - al1;
        yp = z;
        q2c = atan2(yp,xp) - atan2(al3*sin(q3c), al2+al3*cos(q3c));

        qo = [q1c 0; q2c; q3c]*180/pi;
    catch
        warning('unreachable');
    end
% numerical computation
aq1 = q1c; aq2 = q2c(2); aq3 = q3c(2);
cp = double(subs(P03, [q1, q2, q3, l1, l2, l3], [aq1, aq2, aq3, al1, al2, al3]));
M=[M,cp];
        end
end
```

MATLAB Script 8-4, adapted from [38]

```
%quintic polynomial trajectories
qd1=[];qd2=[];qd3=[];
vd1=[];vd2=[];vd3=[];
ad1=[];ad2=[];ad3=[];
tt=[];

for i = 0.0: 1.0: 3.0
%parameters of b (conditions vector)
v0 = 0; v1 = 0;
t0 = i; tf = i+1;
ac0 = 0; ac1 = 0;
t = linspace(t0,tf,100); %time vector (linearly spaced)

%XYZ Coordinates when t=t0 (LEG1)
q1_0 = (abs(0.5*sin(2*pi*0.125*t0))-0.25)*(-1);
q2_0 = ((abs(0.3*sin(2*pi*0.25*t0))+(0.3*sin(2*pi*0.25*t0)))*-0.5)+1;
q3_0 = ((abs(0.5*sin(2*pi*0.25*t0))+(0.5*sin(2*pi*0.25*t0)))*0.5)-1;

%XYZ Coordinates when t=tf (LEG1)
q1_1 = (abs(0.5*sin(2*pi*0.125*tf))-0.25)*(-1);
q2_1 = ((abs(0.3*sin(2*pi*0.25*tf))+(0.3*sin(2*pi*0.25*tf)))*-0.5)+1;
q3_1 = ((abs(0.5*sin(2*pi*0.25*tf))+(0.5*sin(2*pi*0.25*tf)))*0.5)-1;

U = ones(size(t)); %step signal
M =[1   t0  t0^2    t0^3    t0^4        t0^5;
    0   1   2*t0    3*t0^2  4*t0^3      5*t0^4;
    0   0   2       6*t0    12*t0^2     20*t0^3;
    1   tf  tf^2    tf^3    tf^4        tf^5;
    0   1   2*tf    3*tf^2  4*tf^3      5*tf^4;
    0   0   2       6*tf    12*tf^2     20*tf^3];

b1 = [q1_0; v0; ac0; q1_1; v1; ac1]; a1 = M\b1;
b2 = [q2_0; v0; ac0; q2_1; v1; ac1]; a2 = M\b2;
b3 = [q3_0; v0; ac0; q3_1; v1; ac1]; a3 = M\b3;

%position, velocity and acceleration for joint 1
qd1t = a1(1).*U + a1(2).*t+a1(3).*t.^2+a1(4).*t.^3+a1(5).*t.^4+a1(6).*t.^5;
vd1t = a1(2).*U +2*a1(3).*t+3*a1(4).*t.^2+4*a1(5).*t.^3+5*a1(6).*t.^4;
ad1t = 2*a1(3).*U + 6*a1(4).*t+12*a1(5).*t.^2+20*a1(6).*t.^3;

%position, velocity and acceleration for joint 2
qd2t = a2(1).*U + a2(2).*t+a2(3).*t.^2+a2(4).*t.^3+a2(5).*t.^4+a2(6).*t.^5;
vd2t = a2(2).*U +2*a2(3).*t+3*a2(4).*t.^2+4*a2(5).*t.^3+5*a2(6).*t.^4;
ad2t = 2*a2(3).*U + 6*a2(4).*t+12*a2(5).*t.^2+20*a2(6).*t.^3;

%position, velocity and acceleration for joint 3
qd3t = a3(1).*U + a3(2).*t+a3(3).*t.^2+a3(4).*t.^3+a3(5).*t.^4+a3(6).*t.^5;
vd3t = a3(2).*U +2*a3(3).*t+3*a3(4).*t.^2+4*a3(5).*t.^3+5*a3(6).*t.^4;
ad3t = 2*a3(3).*U + 6*a3(4).*t+12*a3(5).*t.^2+20*a3(6).*t.^3;

tt = [tt,t];
qd1 = [qd1,qd1t]; qd2 = [qd2,qd2t]; qd3 = [qd3,qd3t];
vd1 = [vd1,vd1t]; vd2 = [vd2,vd2t]; vd3 = [vd3,vd3t];
ad1 = [ad1,ad1t]; ad2 = [ad2,ad2t]; ad3 = [ad3,ad3t];
end
```

MATLAB Script 8-5, adapted from [16]

```
%lspb (linear segment with parabolic blend)

%initiate with empty matrix for concatenating
qd1=[];qd2=[];qd3=[];
vd1=[];vd2=[];vd3=[];
ad1=[];ad2=[];ad3=[];

for i=0:3
   t0=i;tf=i+1;
   t = linspace(0,1,100); %time vector (linearly spaced)

   %XYZ Coordinates when t=t0 (LEG1)
   q1_0 = (abs(0.5*sin(2*pi*0.125*t0))-0.25)*(-1);
   q2_0 = ((abs(0.3*sin(2*pi*0.25*t0))+(0.3*sin(2*pi*0.25*t0)))*-0.5)+1;
   q3_0 = ((abs(0.5*sin(2*pi*0.25*t0))+(0.5*sin(2*pi*0.25*t0)))*0.5)-1;

   %XYZ Coordinates when t=tf (LEG1)
   q1_1 = (abs(0.5*sin(2*pi*0.125*tf))-0.25)*(-1);
   q2_1 = ((abs(0.3*sin(2*pi*0.25*tf))+(0.3*sin(2*pi*0.25*tf)))*-0.5)+1;
   q3_1 = ((abs(0.5*sin(2*pi*0.25*tf))+(0.5*sin(2*pi*0.25*tf)))*0.5)-1;

   %Calculate lspb
   [q1,v1,a1]=lspb(q1_0,q1_1,t);
   [q2,v2,a2]=lspb(q2_0,q2_1,t);
   [q3,v3,a3]=lspb(q3_0,q3_1,t);

   %concatenate
   qd1=[qd1,q1];qd2=[qd2,q2];qd3=[qd3,q3];
   vd1=[vd1,v1];vd2=[vd2,v2];vd3=[vd3,v3];
   ad1=[ad1,a1];ad2=[ad2,a2];ad3=[ad3,a3];
end
```

MATLAB Script 8-6, the LSPB toolbox can be found in [12]

```
%quintic polynomial trajectories using heuristics

qx=[];qy=[];qz=[];qd1=[];qd2=[];qd3=[];vd1=[];vd2=[];vd3=[];
ad1=[];ad2=[];ad3=[];time=[];

t=[0,0.5,1,1.5,2,2.5,3,3.5,4]; %Time Vector
[m,n] = size(t);k=n-1;

for j = 1:1:n
        %Position Vectors x, y and z
        x = (abs(0.5*sin(2*pi*0.125*t(j)))-0.25)*(-1);
        y = ((abs(0.3*sin(2*pi*0.25*t(j)))+(0.3*sin(2*pi*0.25*t(j))))*-0.5)+1;
        z = double(((abs(0.5*sin(2*pi*0.25*t(j)))+(0.5*sin(2*pi*0.25*t(j))))*0.5)-1);
        qx=[qx,x]; qy=[qy,y]; qz=[qz,z];
end

for i = 1:1:k
vxi=[0,0];vyi=[0,0];vzi=[0,0]; %Velocity Vector

[vx,vy,vz]=vel_heur(vxi,vyi,vzi,qx,qy,qz,t,k); %Calculate Intermediate Vel.

%parameters of b (conditions vector)
t0=t(i);tf=t(i+1); %final time
tt = linspace(t0,tf,100); %time vector (linearly spaced)

U = ones(size(tt)); %step signal
M =[1   t0  t0^2    t0^3    t0^4        t0^5;
    0   1   2*t0    3*t0^2  4*t0^3      5*t0^4;
    0   0   2       6*t0    12*t0^2     20*t0^3;
    1   tf  tf^2    tf^3    tf^4        tf^5;
    0   1   2*tf    3*tf^2  4*tf^3      5*tf^4;
    0   0   2       6*tf    12*tf^2     20*tf^3];

%conditions vector and vector of coefficients
b1 = [qx(i); vx(i);0; qx(i+1); vx(i+1);0]; a1 = M\b1;
b2 = [qy(i); vy(i);0; qy(i+1); vy(i+1);0]; a2 = M\b2;
b3 = [qz(i); vz(i);0; qz(i+1); vz(i+1);0]; a3 = M\b3;
%position, velocity and acceleration for joint 1
qd1t = a1(1).*U + a1(2).*tt+a1(3).*tt.^2+a1(4).*tt.^3+a1(5).*tt.^4+a1(6).*tt.^5;
vd1t = a1(2).*U +2*a1(3).*tt+3*a1(4).*tt.^2+4*a1(5).*tt.^3+5*a1(6).*tt.^4;
ad1t = 2*a1(3).*U + 6*a1(4).*tt+12*a1(5).*tt.^2+20*a1(6).*tt.^3;
%position, velocity and acceleration for joint 2
qd2t = a2(1).*U + a2(2).*tt+a2(3).*tt.^2+a2(4).*tt.^3+a2(5).*tt.^4+a2(6).*tt.^5;
vd2t = a2(2).*U +2*a2(3).*tt+3*a2(4).*tt.^2+4*a2(5).*tt.^3+5*a2(6).*tt.^4;
ad2t = 2*a2(3).*U + 6*a2(4).*tt+12*a2(5).*tt.^2+20*a2(6).*tt.^3;
%position, velocity and acceleration for joint 3
qd3t = a3(1).*U + a3(2).*tt+a3(3).*tt.^2+a3(4).*tt.^3+a3(5).*tt.^4+a3(6).*tt.^5;
vd3t = a3(2).*U +2*a3(3).*tt+3*a3(4).*tt.^2+4*a3(5).*tt.^3+5*a3(6).*tt.^4;
ad3t = 2*a3(3).*U + 6*a3(4).*tt+12*a3(5).*tt.^2+20*a3(6).*tt.^3;
% concatenate
qd1 = [qd1,qd1t];qd2 = [qd2,qd2t];qd3 = [qd3,qd3t];
vd1 = [vd1,vd1t];vd2 = [vd2,vd2t];vd3 = [vd3,vd3t];
ad1 = [ad1,ad1t];ad2 = [ad2,ad2t];ad3 = [ad3,ad3t];
time=[time,tt];
end
```

MATLAB Script 8-7, adapted from [16]

```
%quintic polynomial trajectories using PRM

qd1=[]; qd2=[]; qd3=[];
vd1=[]; vd2=[]; vd3=[];
ad1=[]; ad2=[]; ad3=[];

%Time Vector
t = linspace(0,4,22); %t=[0,0.5,1,1.5,2,2.5,3,3.5,4];
k=21; %Number of intermediate segments

qx=[2,2,2,2,3,4,5,6,7,8,9,9,9,9,9,8,7,6,5,4,3,2]; qy=ones(22);
qz=[1,2,3,4,5,6,7,8,7,6,5,4,3,2,1,1,1,1,1,1,1,1];

for i = 1:1:k
%Velocity Vector
vxi=[0,0]; vyi=[0,0]; vzi=[0,0];

%Calculate Intermediate Velocities
[vx,vy,vz]=vel_heur(vxi,vyi,vzi,qx,qy,qz,t,k);

%parameters of b (conditions vector)
t0=t(i); tf=t(i+1); %initial and final time
tt = linspace(t0,tf,100); %time vector (linearly spaced)

U = ones(size(tt)); %step signal
M =[1   t0  t0^2    t0^3    t0^4        t0^5;
    0   1   2*t0    3*t0^2  4*t0^3      5*t0^4;
    0   0   2       6*t0    12*t0^2     20*t0^3;
    1   tf  tf^2    tf^3    tf^4        tf^5;
    0   1   2*tf    3*tf^2  4*tf^3      5*tf^4;
    0   0   2       6*tf    12*tf^2     20*tf^3];
%conditions vectors and vector of coefficients
b1 = [qx(i); vx(i);0; qx(i+1); vx(i+1);0]; a1 = M\b1;
b2 = [qy(i); vy(i);0; qy(i+1); vy(i+1);0]; a2 = M\b2;
b3 = [qz(i); vz(i);0; qz(i+1); vz(i+1);0]; a3 = M\b3;
%position, velocity and acceleration for joint 1
qd1t = a1(1).*U +
a1(2).*tt+a1(3).*tt.^2+a1(4).*tt.^3+a1(5).*tt.^4+a1(6).*tt.^5;
vd1t = a1(2).*U +2*a1(3).*tt+3*a1(4).*tt.^2+4*a1(5).*tt.^3+5*a1(6).*tt.^4;
ad1t = 2*a1(3).*U + 6*a1(4).*tt+12*a1(5).*tt.^2+20*a1(6).*tt.^3;
%position, velocity and acceleration for joint 2
qd2t = a2(1).*U +
a2(2).*tt+a2(3).*tt.^2+a2(4).*tt.^3+a2(5).*tt.^4+a2(6).*tt.^5;
vd2t = a2(2).*U +2*a2(3).*tt+3*a2(4).*tt.^2+4*a2(5).*tt.^3+5*a2(6).*tt.^4;
ad2t = 2*a2(3).*U + 6*a2(4).*tt+12*a2(5).*tt.^2+20*a2(6).*tt.^3;
%position, velocity and acceleration for joint 3
qd3t = a3(1).*U +
a3(2).*tt+a3(3).*tt.^2+a3(4).*tt.^3+a3(5).*tt.^4+a3(6).*tt.^5;
vd3t = a3(2).*U +2*a3(3).*tt+3*a3(4).*tt.^2+4*a3(5).*tt.^3+5*a3(6).*tt.^4;
ad3t = 2*a3(3).*U + 6*a3(4).*tt+12*a3(5).*tt.^2+20*a3(6).*tt.^3;
%concatenate
qd1 = [qd1,qd1t];qd2 = [qd2,qd2t];qd3 = [qd3,qd3t];
vd1 = [vd1,vd1t];vd2 = [vd2,vd2t];vd3 = [vd3,vd3t];
ad1 = [ad1,ad1t];ad2 = [ad2,ad2t];ad3 = [ad3,ad3t];
end
```

MATLAB Script 8-8, adapted from [16]

Hexapod Modelling, Path Planning, and Control

Third Year Individual Project – Progress Report

Nov 2016

Canberk Suat Gurel

9023527

Supervisor: Dr Joaquin Carrasco Gomez

Contents

8.4.1. Introduction

The core of this project is to obtain a better understanding of how to ameliorate the locomotion speed while maintaining the stability of the robot. A number of factors that play a vital role in the robot will be deliberated such as walking gaits, computation of end effector trajectory, motion planning strategies and navigation control systems. The project is primarily focused on the simulations in which MapleSim is used as simulation platform however, a single hexapod robot leg is intended to be manufactured and programmed for demonstration purpose. Two sets of hexapod models with different body shapes - one is rectangular and the other is circular - are created on MapleSim, both of the models are operating using inverse kinematics. The mentioned factors will be investigated in depth along with tilt control and curved locomotion by implementing them in the MapleSim models. Then, the simulation results will be analysed and reported in a cogent manner.

This progress report addresses the outlines of the final report, elucidates the aims and objectives of the project and then elaborates the progress of the project in which the model design procedure has been briefly identified. The necessity of using the inverse kinematics is explained and the step by step derivation of the equations has been delivered. The results that are obtained from the initial experiments regarding to the tripod gait and how the locomotion speed is increased by the implementation of an alternative stepping type are demonstrated. Furthermore, the problems that have been encountered are delivered along with the solutions found.

8.4.2. Aims and Objectives

The aim of this project is to develop a hexapod robot model that travels from point A to B, in the fastest time possible by calculating shortest path and avoiding the obstacles that are located on an uneven terrain. The stages along the way to accomplish the aim are as follows:

- Develop circular and rectangular body shaped hexapod models in the MapleSim environment by implementing inverse kinematics to control the leg movements.
- Investigate the effects of different walking gaits – tripod, ripple and wave – and stepping types - rectangular, triangular and bell shaped - on the stability and locomotion speed by considering the force estimation of each leg.
- Develop a model which climbs stairs by keeping the link 3 of its legs perpendicular to the ground.

- Implement bang-bang, and PI controllers to make the model follow a straight line then make it follow a pre-determined curve using the chosen controller.
- Calculate the shortest path that needs to be followed in order to get from point A to point B by using the A* method then apply the A* method into the model to make it follow the calculated path on an uneven terrain.
- Investigate how to recover in case of the failure of one or more legs.

8.4.3. Progress to Date

Literature Review

An in depth research of the existing projects regarding to the different implementations of motion planning and control strategies is absolutely crucial in order to expand the perspective and develop a better understanding of alternative concepts. [39]

There are a number of concurrent factors that need to be taken into account while calculating the motion planning of a multi legged robot. Firstly, it is quite difficult to distinguish the obstacles that could be passed over without getting marginally close to it. Secondly, the working envelopes of each leg and the revolution restriction due to mechanical constraints of each joint should be considered. Thirdly, the robot has a certain stability limitation which needs to be ensured that it will not be exceeded while overcoming a particular obstacle. [14]

Bai and Low [15], adverted a method in which the terrain is mapped by using potential fields to classify the obstacles into two categories depending on whether they could be passed over or not. Then, the path that needs to be followed is obtained by BFP method. A similar approach is expressed by Wooden [40], a hydraulically actuated robot – BigDog - that is engineered by Boston Dynamics uses a combination of LIDAR which is based on planer laser scan and Stereo vision system that is based on image processing to generate the 3D map of the terrain. Then, the path is calculated by the A* method. Both of these methods will be covered in the final report. Kolter [8], mentioned about another product of Boston Dynamics, the LittleDog, which operates in the same manner. Initially, the 3D map of the terrain is generated in order to deduce the obstacles that are high enough to be collided. Secondly, the standard deviation of the mapped terrain is examined to deduce the local maximums and minimums of the terrain. Then, this information is used to create the cost map which indicates where each leg should be positioned and the trajectory

of leg end effectors that will be followed between the given leg potions is computed. Finally, the same procedure is repeated to plan the motion of the body. A different approach that consists of 2 phase computation is illuminated by Vernaza [29]. Unlike the last example, this robot considers the terrain as a whole and computes the footstep trajectory by applying global planning and then, undertakes a second computation that works out the angle of rotation of each joint that in order to follow the given step trajectory. Then, the R* algorithm is applied, which will also be covered in the final report. Kalakrishnan [18] suggested an alternative which has a more complex control architecture [14]. Initially, the rough terrain is mapped and an approximation of the path that will be followed by the body is made. Then, the control architecture runs within a continuous loop which executes once per footstep at a frequency of 100Hz. Firstly, the footstep planner is performed to deduce the next 4 (6 for a hexapod) positions of the leg end effectors. Secondly, the pose finder is used to correct the pose of the robot in order to improve the work envelop of the legs while performing obstacle avoidance. Thirdly, the body and foot trajectory planners operate to determine the collision free route that needs to be followed by legs and land on the projected positions. Then, the same procedure is repeated until it is terminated by the operator. A different path planning method, rapidly exploring random trees (RRT) method, has been asserted by LaValle [13] which is fairly useful for both holonomic and nonholonomic systems and high degrees of freedom (up to 12 DoF). The details of RRT method will be covered in the final report.

Model Design Procedure

In the initial discussion with the project supervisor it was stated that although the general project outline has been established, the project content could be formed according to the interests of the student.
In the following meetings, an agreement has been reached that the inverse kinematics will be implemented, having said that, it is possible to move the model by simply inputting the rotation angle of the revolute joint at a given time. This rather primitive method has been used in the hexapod model that was provided by the Maplesoft, this model could be considered as a good starting point however, a more advance method i.e. inverse kinematics was necessary to implement autonomous navigation which will be further investigated in the final report.
Furthermore, it has been decided in the first several weeks that the main focus of the project would be the navigation control system of the hexapod robot and the project would be based on MapleSim simulations. The main reason behind primarily

focusing on computer simulations instead of a physical model is that by doing so, all the procedures from design to manufacturing and assembly could be bypassed and the path planning and control strategies could be dived straight in. As a result of excessive amount of library research and going through over a dozen of books, an understanding of the materials that will be covered in the final report has been acquired. Additionally, the regular meetings with the project supervisor have provided guidance through the project and enlightened the pathway that needs to be pursued.

8.4.4. Theoretical Development

As the complexity of the system increases, the robot is desired to undertake as many calculations as possible since the computation capabilities of human operator are limited. [14] Similarly, calculating the angle of joint rotation for 6 legs in order to move the hexapod robot from position A to position B is a challenging task for the operator therefore, the robot is required to compute this calculation and be capable of moving the end effector of the robot legs to the given coordinates. This task is handled by inverse kinematics equations in which the joint angles are calculated that will make the end effector achieve the particular X, Y, and Z coordinates. [41] The derivation of equations are as follows:

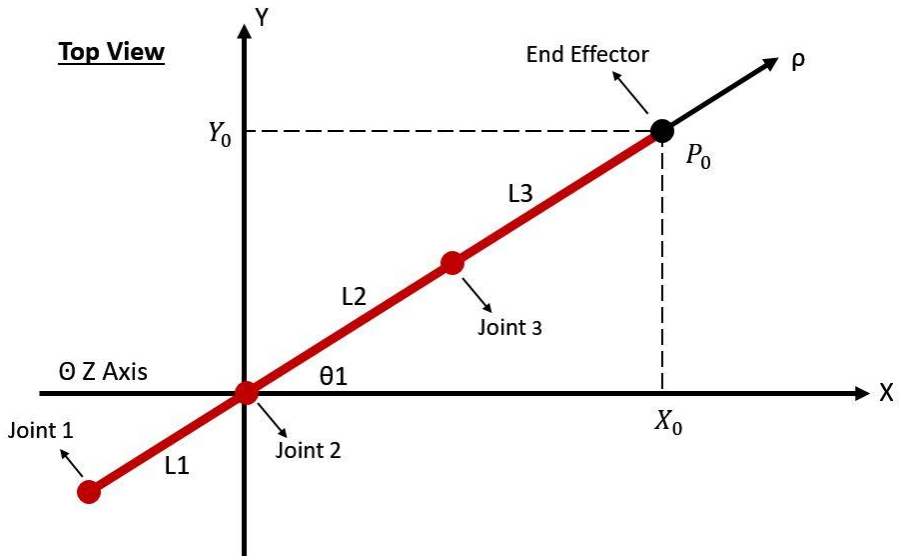


Figure 3.3.1 Top view of a single robot leg

Figure 3.3.1 shows the top view of a single robot leg. An axis is defined in the direction of the leg and it is labelled as **ρ.** The coordinates of the end effector are (X_0,Y_0) and the angle of rotation about the Z axis is θ1.

The origin has been defined as the centre of the Joint 2 since the θ1 is independent of where the origin of the leg is defined due to Z rule in geometry. In other words,

defining the Joint 1 as the origin of the robot leg would simply add an offset of L1 to the system. The θ1 is calculated by Equation 1:

$$\theta 1 = \arctan\left(\frac{Y_0}{X_0}\right) \quad \text{.....(1)}$$

The distance between the origin and the end effector is expressed as:

$$P_0 = \sqrt{{X_0}^2 + {Y_0}^2} \quad \text{.....(2)}$$

Figure 3.3.2 and 3.3.3 demonstrate the side view of a single leg.

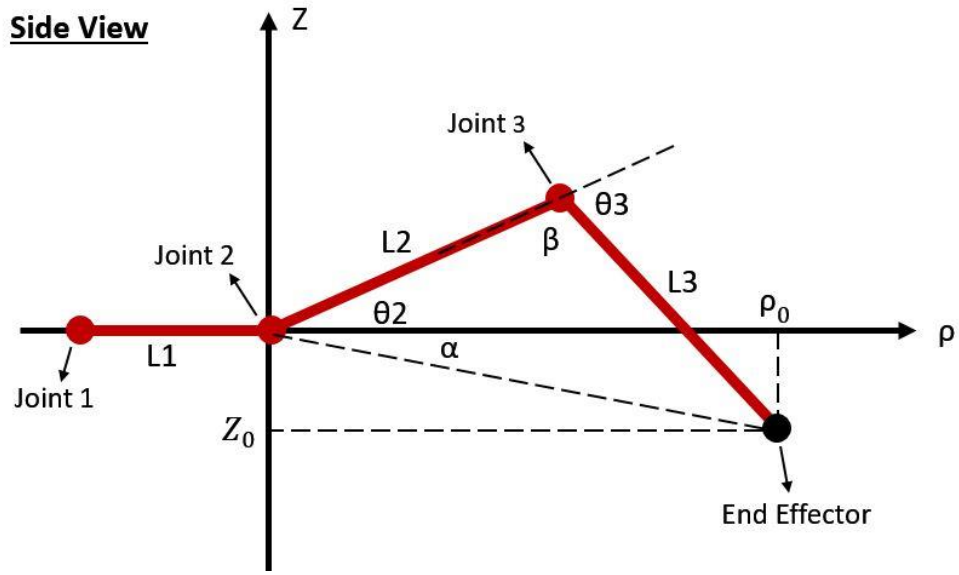


Figure 3.3.2 Side view of a single robot leg

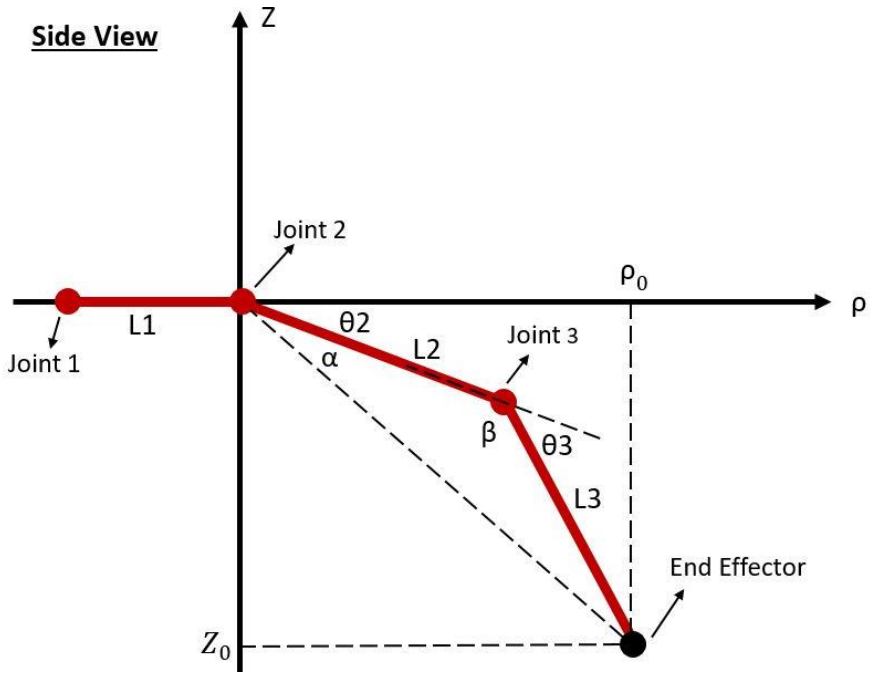


Figure 3.3.3 Side view of a single robot leg

The coordinates of the end effector are (ρ_0,Z_0). The distance between the origin and

the end effector is $\sqrt{{\rho_0}^2 + {Z_0}^2}$ ……(3)

The angle between L2 and L3 is labelled as β and it is computed by applying law of cosine.

$$\beta = \arccos\left(\frac{{Z_0}^2+{\rho_0}^2-L2^2-L3^2}{-2*L2*L3}\right) \quad \text{……..…….. (4)}$$

Equation 5 demonstrates the expression for the angle α.

$$\alpha = \arctan\left(\frac{Z_0}{P_0}\right) = \arctan\left(\frac{Z_0}{\sqrt{{X_0}^2+{Y_0}^2}}\right) \quad \text{…….……(5)}$$

The θ2 is calculated by applying the law of cosine which indicates that

$$\theta 2 + \alpha = \arccos\left(\frac{L3^2-L2^2-({\rho_0}^2+{Z_0}^2)}{-2*L2*\sqrt{{\rho_0}^2+{Z_0}^2}}\right) \quad \text{……..……..(6)}$$

The angle of rotation of Joints 1, 2, and 3 are:

- $\theta 1 = \arctan\left(\frac{Y_0}{X_0}\right)$
- $\theta 2 = \arccos\left(\frac{-L3^2+L2^2+{X_0}^2+{Y_0}^2+{Z_0}^2}{2*L2*\sqrt{{X_0}^2+{Y_0}^2+{Z_0}^2}}\right) + \arctan\left(\frac{Z_0}{\sqrt{{X_0}^2+{Y_0}^2}}\right)$
- $\theta 3 = -(\pi - \beta) = -\arccos\left(\frac{{X_0}^2+{Y_0}^2+{Z_0}^2-L2^2-L3^2}{2*L2*L3}\right)$

8.4.5. Practical Development

The hexapod model consists of 6 legs and each leg has 3 revolute joints. Figure 3.4.1 shows a rectangular body shaped hexapod model and the red arrow specifies the direction of motion. Note that, the Z axis is pointing out from the paper.

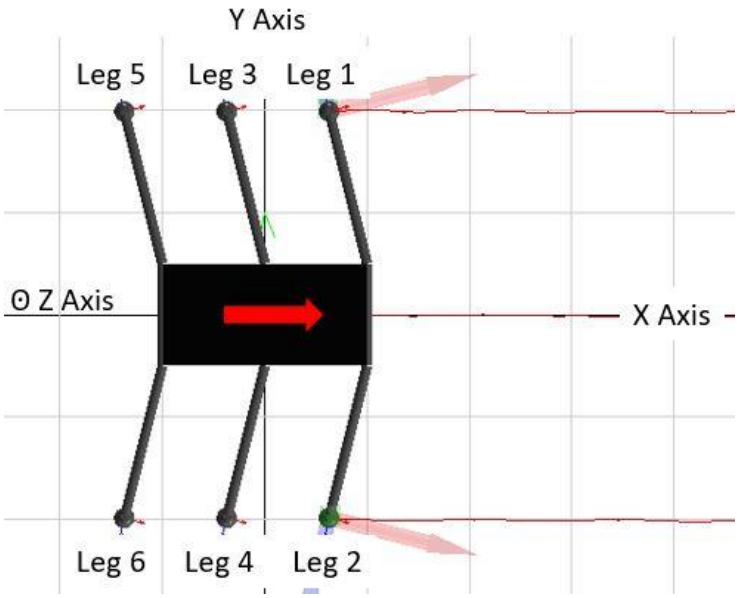


Figure 3.4.1 Top View of the Rectangular Body Shaped Hexapod

The angles θ1, θ2 and θ3 describes the angle of rotation of joints 1, 2 and 3 respectively. The joint 1 allows the rotation motion about Z-axis, which moves the leg forwards and backwards. The other 2 joints, are defined to rotate about the X- axis, which moves the leg up and down. Figure 3.4.2 illustrates a single leg of the hexapod model.

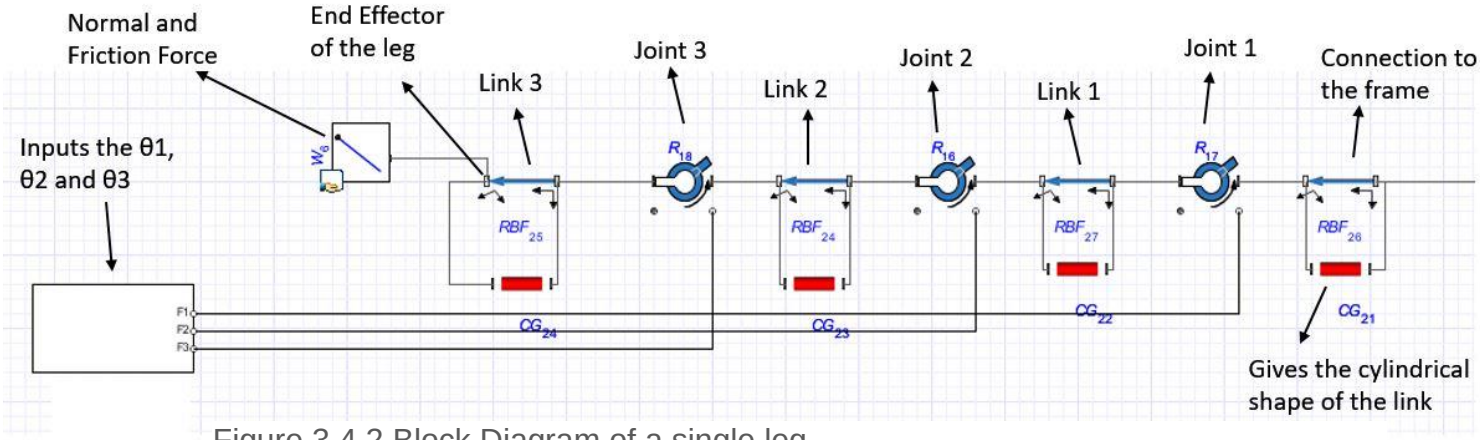


Figure 3.4.2 Block Diagram of a single leg

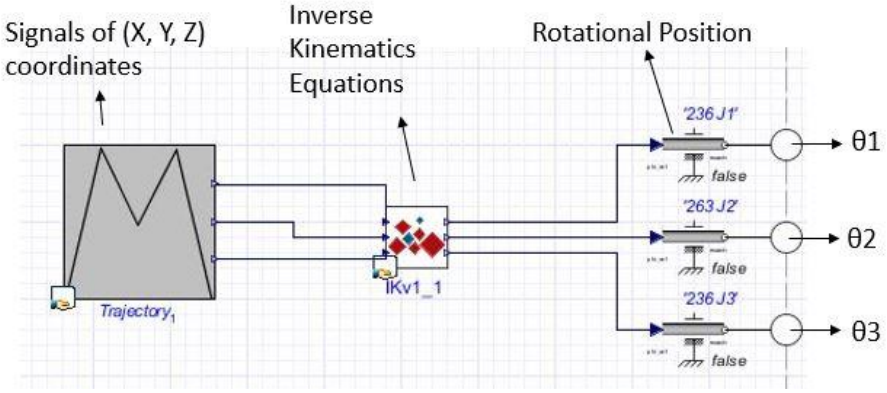


Figure 3.4.3 Block diagram of joint inputs

The X, Y, Z coordinates of the end effector are fed into the inverse kinematics block which calculates the angle of rotations of joints 1-2-3 and then the computed angles are applied to the corresponding revolute joint in order to move the end effector to the desired coordinates. Figure 3.4.3 demonstrates the input of the joints 1-2-3.

The leg is moved forward by altering the input signals which indicate the position of the end effector. The robot walks forward by moving legs in synchronous with each other, which is done by adding a time offset to signals. There are 3 commonly used walking gaits – tripod, ripple and wave gaits - that are produced by applying different time offsets to the input signals [4]. Figure 3.4.4 [17] shows the stance and swing phases of the tripod gait. The solid lines represent the swing phase of the leg whereas the dashed lines are the stance phase of the leg. Only tripod gait will be

covered in the progress report, the other two will be evaluated in the final report.

Since, the position of joint 2 is defined as the origin of each leg, the identical input signal which indicates the X, Y and Z coordinates can be applied to each one of the 6 legs only by adjusting the time offsets according to the chosen walking gait. For example, the tripod gait could be implemented by applying 2 sets of trajectory signals, which are apart from each other by a time offset, to the Legs 2-3-6 and Legs 1-4-5 respectively. The Y coordinates of the leg end effectors are kept constant. The Y coordinates need to be altered in order to follow a curved pathway.

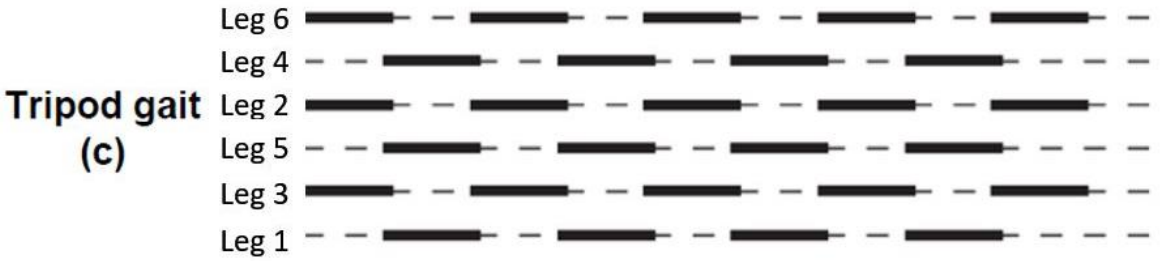


Figure 3.4.4 Stance and swing phases of legs for the tripod gait [17]

Figure 3.4.5 (in the appendix) demonstrates the sequence of leg motions which needs to be performed in order to obtain the tripod gait. Since, none of the Up, Forward, Down and Backward phases are performed simultaneously, the waveform that is displaying the end effector footstep is expected to be in the form of a rectangle. However, the end effector footstep trajectory is in the trapezoid shape rather than a rectangular due to inertia. Figure 3.4.6 shows the trajectory of the end effector.

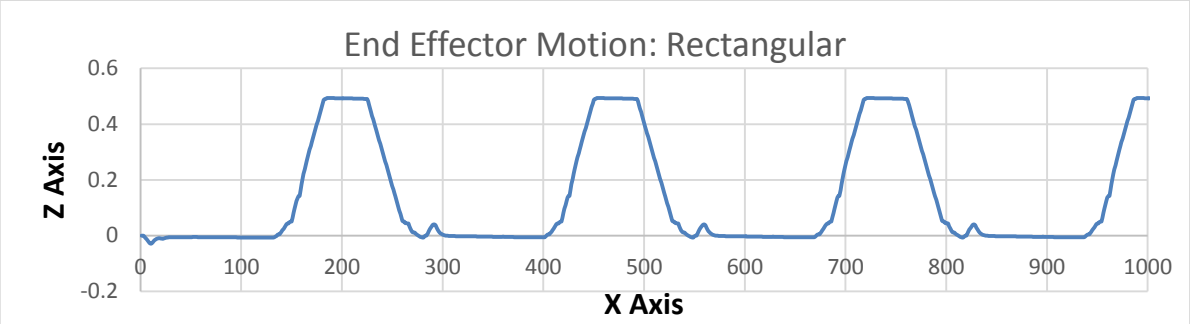


Figure 3.4.6 Side view of the path that is followed by the end effector of a single leg

The sequence of leg motions that are displayed in Figure 3.4.5 (in the appendix) can be slightly altered for optimisation. It is possible to change the footstep type from rectangular to triangular by applying some of the signals simultaneously. Figure 3.4.7 (in the appendix), illustrates the sequence of leg motions that generates the tripod gait with triangular footsteps. Figure 3.4.8 shows the motion of the end effector when

the locomotion sequence that is displayed on Figure 3.4.7 is applied. Comparing Figures 3.4.6 and 3.4.8, it can be realised that, the hexapod model has taken more steps within the same time duration, which indicates that the locomotion speed is improved. It is also possible to obtain a bell-shaped footstep by applying a sinusoidal signal. In this report, only rectangular and triangular steps are covered, the bell-shaped footstep will be included in the final report.

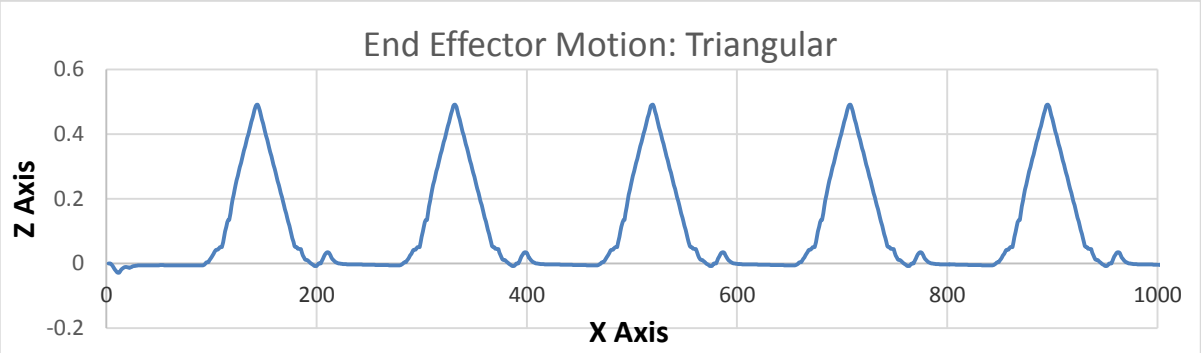
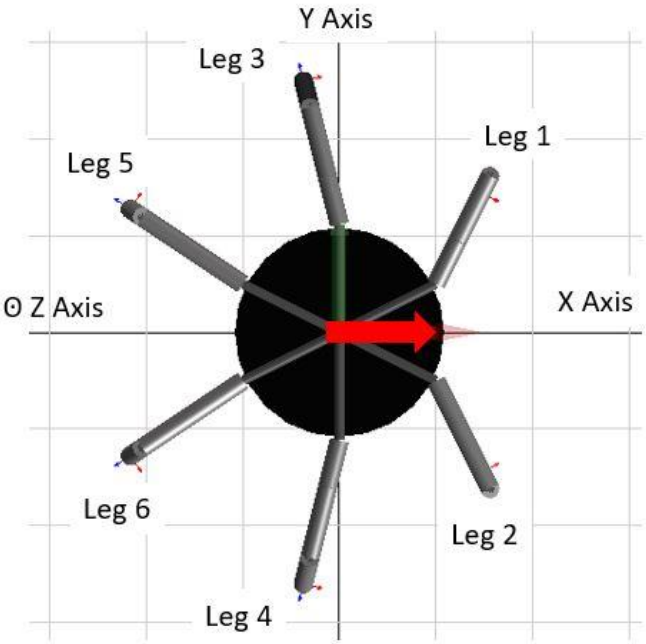


Figure 3.4.8 Side view of the path that is followed by the end effector of a single leg



Additionally, since the legs have been spread out equally the model is more stable. Figure 3.4.9 shows the circular body shaped hexapod robot.

The circular shaped hexapod model locomotion operates in a similar manner however, the front 2 legs and the rear 2 legs are positioned closer to each other which provides 2 advantages to this model. [42] The legs of this configuration are capable of moving within a wider work envelope which is the range of area that the leg can reach. Therefore, the circular shaped hexapod is capable of changing its direction quicker in comparison to the rectangular hexapod configuration.

Figure 3.4.9 Top view of the circular body shaped hexapod model

8.4.6. Problems Encountered

Throughout the modelling procedure, several unforeseen problems have been encountered which caused a delay in the project timeline. As the complexity of the

model increases the number of equations that needs to be computed rises exponentially. As a result, the time taken for the simulation to undertake the equations has peaked and the software started crashing while the visualisation data was being generated. It was a challenging process to find the reason lying behind this case. In order to avoid over complicating the robot locomotion, the smallest increment of one period of the input signals has been set to 1. To give an illustration, Figure 3.4.5 should be examined, the Up, Forward and Down phases each take 1 second whereas the Backward phase takes 3 seconds. In other words, 1 step is taken in every 6 seconds and only 16.67 steps are taken by the end of default simulation period which is 100 seconds. Increasing the simulation period from 100s to 300s should have risen the range of the locomotion however, due to increased number of frames that needs to be generated, the software has failed. By decreasing the fps of the simulation from 60 to 24, the problem has been solved. Alternatively, the time taken for a single step could be reduced and the period of the simulation could have been kept within 100 seconds. Another problem that caused a delay in the schedule was due to the version of the MapleSim software. A number of models that are provided online were generated on the latest version of MapleSim and these models which would potentially improve the project could not be accessed yet.

8.4.7. Further Work Required

In the next stage of the project, the hexapod model will be modified to make it follow a curved path in which a motion controller will be used to ensure that the model stays on the desired pathway. Moreover, 3 different path planning methods that are mentioned in the literature review section will be implemented in the model to determine which one operates better. Eventually, the model is expected to navigate on an uneven terrain by working out the shortest path that needs to be followed in order to reach the final destination.

8.4.8. Conclusion

A summary of the initial steps taken over the past weeks to achieve the desired project aim is provided in this progress report. To sum up, the existing path planning and control strategies have been researched, 3 different walking gaits and footstep types are simulated, inverse kinematics equations are derived and implemented along with 2 different motion controllers in the MapleSim environment. Although, a number of difficulties have been encountered, the major ones have been solved eventually by approaching in a systematic manner and not giving up. The project is

proceeding on determined project plan and it is expected to be completed within the planned delivery date by accomplishing the identified project aim.

8.4.9. Appendix

Simulation Visuals

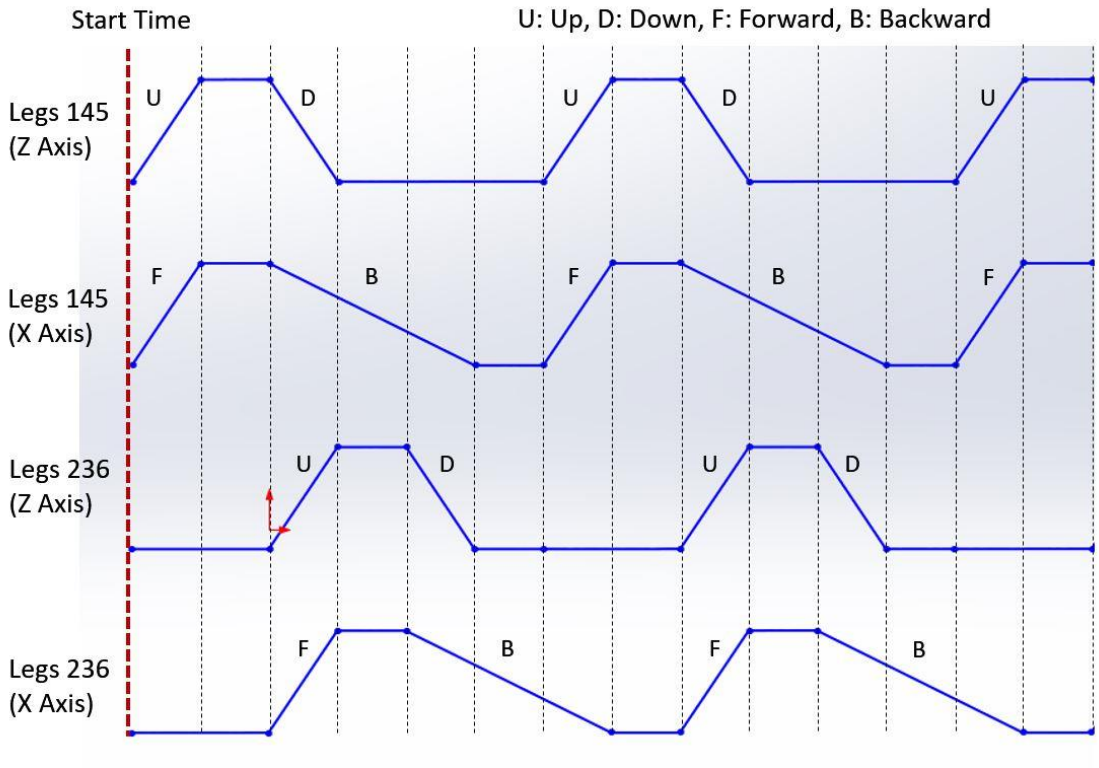


Figure 3.4.5 Locomotion sequence for tripod gait and triangular footsteps

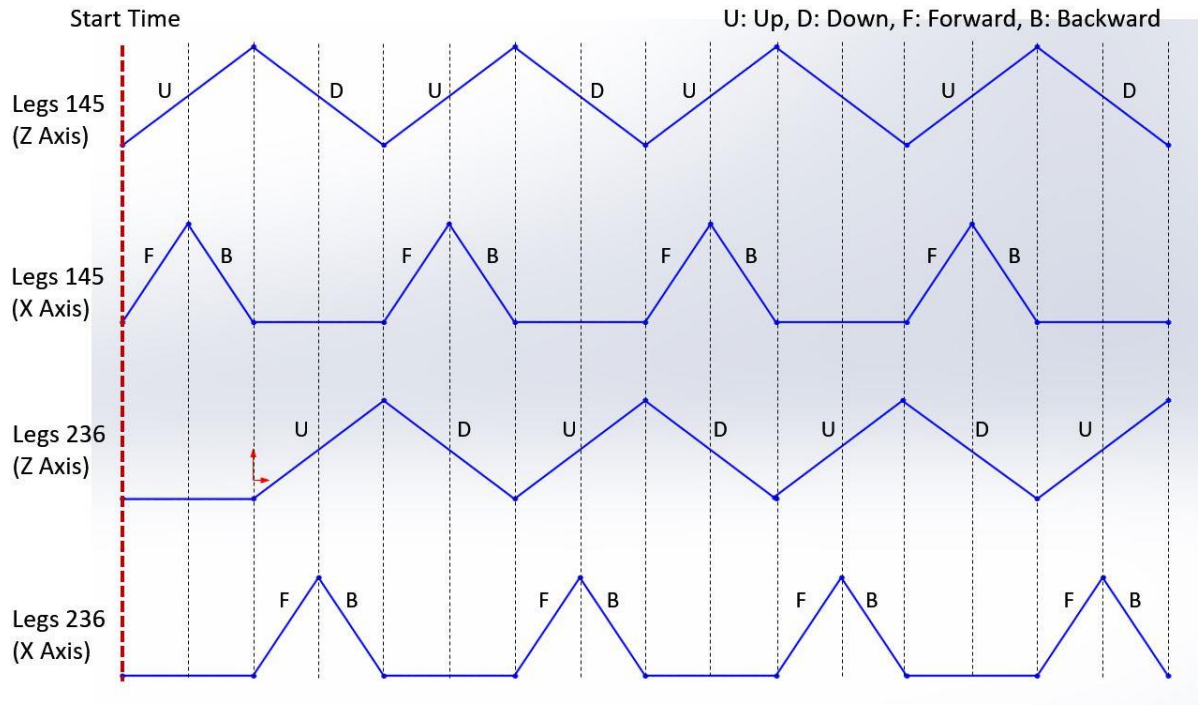


Figure 3.4.7 Locomotion sequence for tripod gait and rectangular footsteps

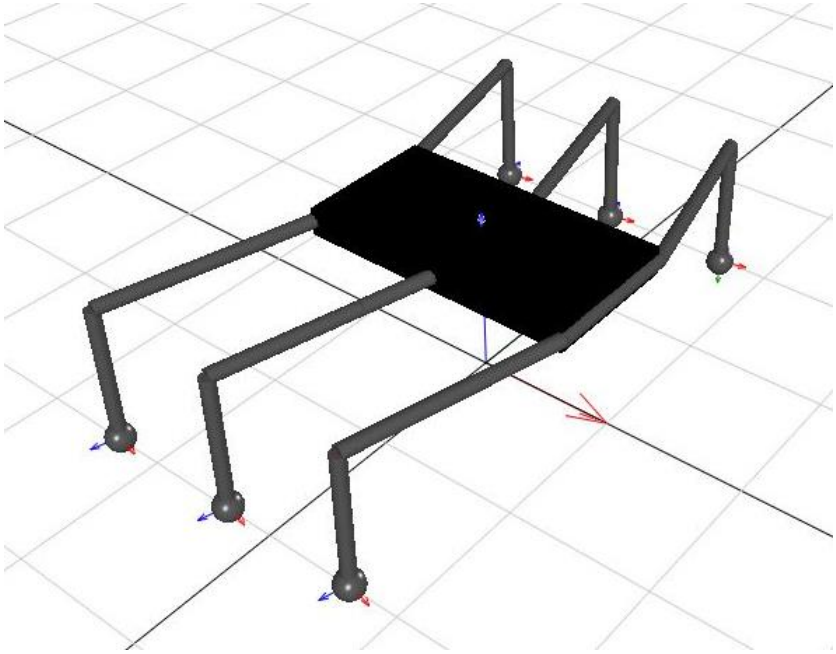


Figure 8.1.1 Rectangular Body Shaped Hexapod Model

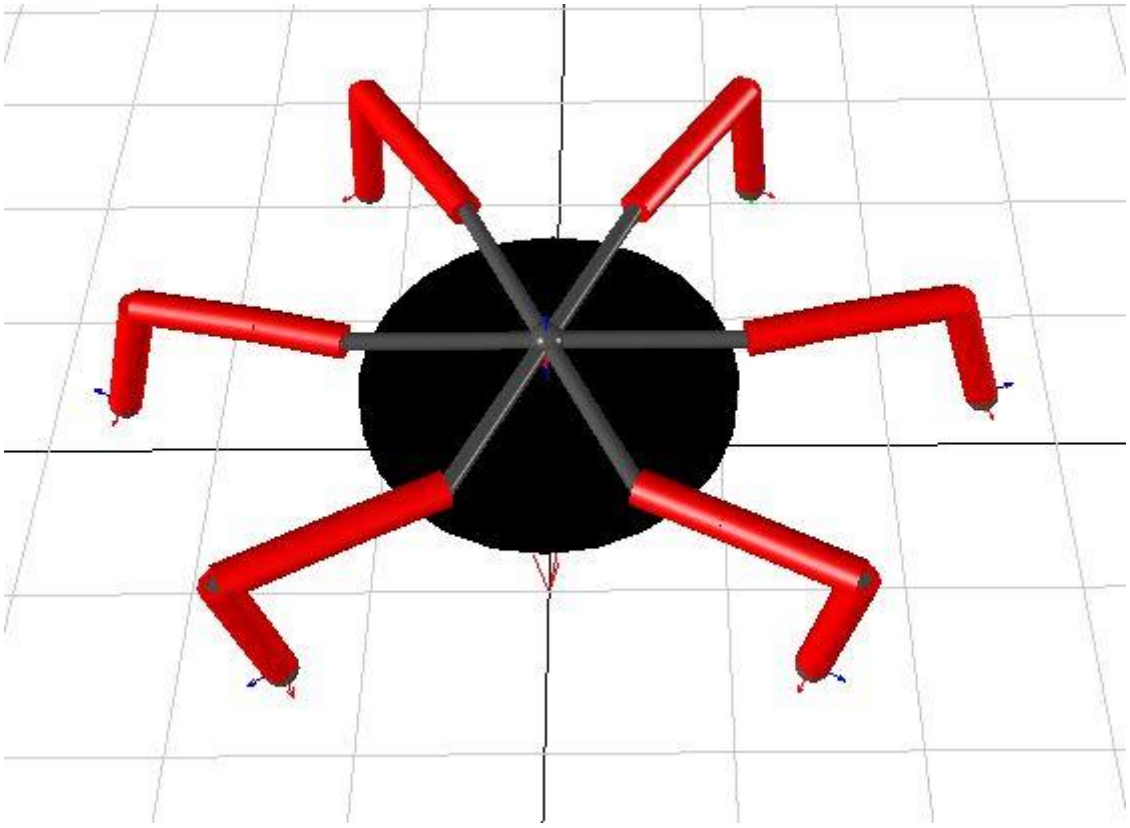


Figure 8.1.2 Circular Body Shaped Hexapod Model

Trigonometry

Law of cosine:

$$a^2 = b^2 + c^2 - 2bc * \cos(\alpha)$$ [43]

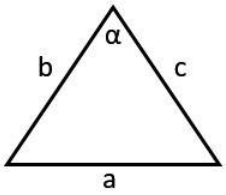
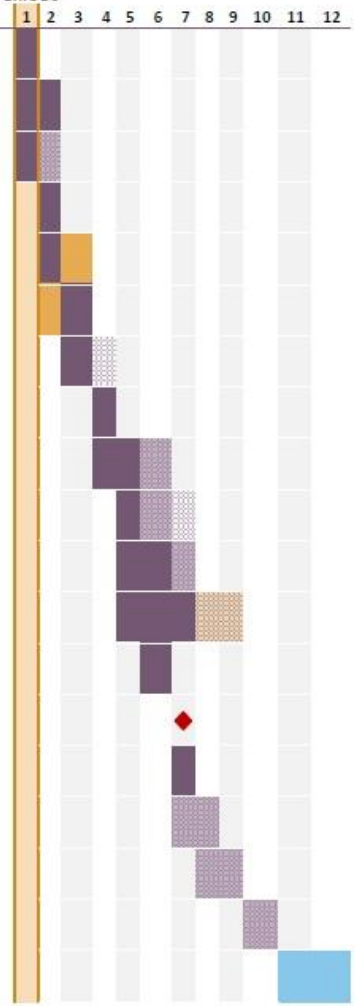


Figure 8.2.1 Law of cosine

8.4.10. Project Plan

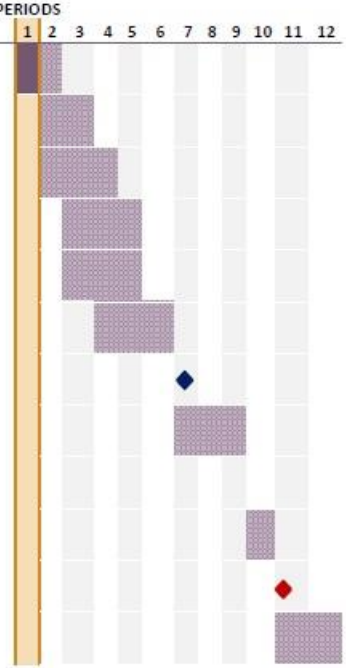
Project Plan: Semester 1

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Library research for resources related to hexapod robots, path planning and control	1	1	1	1	100%
Research for navigation control system, hierarchical control, and robot locomotion	1	2	1	2	100%
Structure the contents of the final report and set the objectives by reviewing the library resources	1	2	1	2	90%
Develop **rectangular** body shaped hexapod model in the MapleSim environment	2	1	2	1	100%
Develop **circular** body shaped hexapod model in the MapleSim environment	2	1	2	2	100%
Implement inverse kinematics to control the leg movements	3	1	2	2	100%
Investigate the effects of different walking gaits on the stability and locomotion speed	3	2	3	1	85%
Make a start with the progress report, review the project handbook and understand the requirements	4	1	4	1	100%
Write parts of the progress report	4	3	4	3	80%
Apply different stepping types in order to speed up the chosen hexapod gait	5	3	5	2	90%
Implement bang-bang, proportional, and PI controllers to make the model follow a straight line	5	3	5	3	70%
Develop a model which climbs stairs by keeping the link 3 of its legs perpendicular to the ground	5	3	5	5	70%
Finalise the progress report and be prepared for submission	6	1	6	1	80%
Milestone: Submission of progress report (07.11.2016)					
Calculate the force estimation of each leg at a given walking gait	7	1	7	1	20%
Make the hexapod follow a pre-determined curve by using the chosen navigation controller	7	2	7	2	0%
Implement the tilt control to the model	8	2	8	2	0%
Contengency week, in case of an unforeseen delay	10	1	10	1	0%
Revision For Exams	0	0	0	0	0%



Project Plan: Semester 2

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research for A*, BFP and RRT methods	1	2	1	2	20%
Implement the these methods to calculate the shortest path that needs to be followed in order to get from point A to point B	2	2	2	2	0%
Design, manufacture, assemble and program the robot leg	2	3	2	3	0%
Apply the RRT path planning concept into the model to make it follow the calculated path on an uneven terrain	3	3	3	3	0%
Investigating how to recover - maintain the stability - if one or more legs fail while moving forward	3	3	3	3	0%
Write parts of the progress report	4	3	4	3	0%
Personal (Unofficial) Milestone: Complete the 1st draft of the final report					
Improve the draft of the final report	7	3	7	3	0%
Easter Break (between weeks 9 and 10)					
Finalise the final report and be prepared for submission	10	1	10	1	0%
Milestone: Submission of Final Report (01.05.2017)					
Oral Examination and Poster	11	2	11	2	0%



8.4.11. Risk Assessment

MANCHESTER 1824

The University of Manchester

SCHOOL OF E&EE RISK ASSESSMENT PRO-FORMA

Date:	*Review Date:*	*Assessed by:*	*Validated by:*	*Location:*	*Reference Number:*
30/10/2016	**[]**			**SSB/[C34]**	**Control_SSB_[location]**
Rig Owner:		*Experiment:*		*Contact Details:*	
[responsible Academic]		**[Developing a single hexapod robot leg]**		**[canberk.gurel@student.manchester.ac.uk]**	

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUPS)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	EXISTING CONTROLS				MEASURE REQUIRED TO PREVENT OR REDUCE RISK (WHAT NEEDS TO BE DONE TO MAKE THE ACTIVITY AS SAFE AS POSSIBLE)	PERSON RESPONSIBLE FOR ACTIONS AND AGREED TIMESCALES TO ACHIEVE THEM	NEW CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE			SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE
Development of a robot leg	Mechanical assembly: screwing, stripping and snipping wires	Finger cuts, jammed finger	The person who is building the robot leg	To use hand protection	1	3	3	Y	To be more careful	Canberk Gurel 2016-2017 Academic Year	1	2	2	Y
Programming of the robot leg	Using computer for a long time	Neck, back injury Visual Impairment Headache	The person who is programmin g the robot leg		1	4	4	Y	To give regular breaks and to use an orthopaedic chair	Canberk Gurel 2016-2017 Academic Year	1	3	3	Y
Testing of the robot leg	The robot leg may hit the objects around it and get damaged	Minor injuries and delay in the project schedule	Surrounding objects and the person who is testing	Mount the leg to a secure environment	1	2	2	Y	Restrict the PWM duty cycle that will be applied	Canberk Gurel 2016-2017 Academic Year	1	2	2	Y
Usage of electrical equipment	Electrical shock, and fire hazard	Burns	Anyone who is around	Electrostatic mat and fire extinguisher	4	1	4	Y	To be more careful	Canberk Gurel 2016-2017 Academic Year	4	1	4	Y

THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (MAX 12 MTHS)

Delay to receive the hardware components	Delay in the schedule and affects the overall performance	Loss of marks	The person who is undertaking the project		3	3	9	Y	Place an order as early as possible	Canberk Gurel 2016-2017 Academic Year	2	2	4	Y
Psychological discomfort	Felling depressed	Loss of energy and difficulty with concentration	The person who is undertaking the project	Inform the project supervisor and welfare officer	3	2	6		Take a break and spend some time elsewhere	Canberk Gurel 2016-2017 Academic Year	3	1	3	Y
Financial discomfort	Exceeding the project budget	Loss of marks	The person who is undertaking the project	Some suppliers are less expensive	3	1	3	Y	Better allocate the budget	Canberk Gurel 2016-2017 Academic Year	3	1	3	Y
Delay in receiving technical support	Delay in the timeline of the project	Loss of marks	The person who is undertaking the project	Inform the project supervisor and the head of technicians	3	1	3	Y	Allow adequate amount of time to receive a response	Canberk Gurel 2016-2017 Academic Year	3	1	3	Y

RISK ASSESSOR:

signed: _____

MANAGER:

signed: _____

IF THE ANSWERS TO ANY OF THE QUESTIONS BELOW IS YES THEN ADDITIONAL SPECIFIC RISK ASSESSMENTS MAY BE REQUIRED.

IS THERE A RISK OF FIRE?	**N**	DOES THE ACTIVITY REQUIRE ANY HOME WORKING?	**N**
ARE SUBSTANCES THAT ARE HAZARDOUS TO HEALTH USED?	**N**	ARE THE EMPLOYEES REQUIRED TO WORK ALONE	**N**
IS THERE MANUAL HANDLING INVOLVED?	**N**	DOES THE ACTIVITY INVOLVE DRIVING	**N**
IS PPE WORN OR REQUIRED TO BE WORN?	**N**	DOES THE ACTIVITY REQUIRE WORK AT HEIGHT	**N**
ARE DISPLAY SCREENS USED?	**Y**	DOES THE ACTIVITY INVOLVE FOREIGN TRAVEL	**N**
IS THERE A SIGNIFICANT RISK TO YOUNG PERSONS?	**N**	IS THERE A SIGNIFICANT RISK TO NEW / PREGNANT MOTHERS?	**N**

THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (MAX 12 MTHS)

The University of Manchester

SEVERITY VALUE = **Potential consequence of an incident/injury given current level of controls.**

5 Very High Death / permanent incapacity / widespread loss
4 High Major Injury (Reportable Category) / Severe Incapacity / Serious Loss
3 Moderate Injury / illness of 3 days or more absence (reportable category) / Moderate loss
2 Slight Minor injury / illness - immediate 1st Aid only / slight loss
1 Negligible No injury or trivial injury / illness / loss

LIKELIHOOD = **what is the potential of an incident or injury occurring given the current level of controls.**

5 Almost certain to occur
4 Likely to occur
3 Quite possible to occur
2 Possible in current situation
1 Not likely to occur

Likelihood \ Severity	1	2	3	4	5
1	Low	Low	Low	Low	Low
2	Low	Low	Medium	Medium	Medium
3	Low	Medium	Medium	High	High
4	Low	Medium	High	High	High
5	Low	Medium	High	High	High

The intersection of the chosen column with the chosen row is the Risk classification.

RISK SCORE Key:

1- 5 LOW - Tolerable - Monitor and Manage

6 - 10 MEDIUM - Review and introduce additional controls to mitigate to ALARP

12 - 25 HIGH - Intolerable Stop Work and immediately introduce further control measures

THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (MAX 12 MTHS)

9. References

[1] M. Gambino, “A Salute to the Wheel,” SMITHSONIAN.COM, 17 06 2009. [Online]. Available: https://goo.gl/wMs0nG. [Accessed 26 04 2017].

[2] K. Nonami, R. K. Barai, A. Irawan and M. R. Daud, Hydraulically Actuated Hexapod Robots: Design, Implementation and Control, Japan: Springer International Publishing, 2014.

[3] P. G. d. Santos, E. Garcia and J. Estremera, Quadruped Locomotion: An Introduction to the Control of Four-legged Robots, Germany: Springer International Publishing, 2005.

[4] M. Zak, J. Rozman and F. V. Zboril, “Ovieview of Bio-Inspired Control Mechanism for Hexapod Robot,” MIR Labs, USA, 2016.

[5] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada and B. Wilcox, “Motion Planning for Legged Robots on Varied Terrain,” SAGE Publications - The International Journal of Robotics Research, Los Angeles, London, New Delhi and Singapore, 2008.

[6] R. Volpe and K. Kawasaki, “NASA - Jet Propulsion Laboratory,” Caltech, [Online]. Available: https://www-robotics.jpl.nasa.gov/systems/system.cfm?System=11. [Accessed 26 04 2017].

[7] A. Roennau, G. Heppner, M. Nowicki and R. Dillmann, “LAURONV: A Versatile Six-Legged Walking Robot with Advanced Maneuverability,” in *IEEE/ASME International Conference*, Besançon, France, 2014.

[8] J. Z. Kolter, M. P. Rodgers and A. Y. Ng, “A Control Architecture for Quadruped Locomotion Over Rough Terrain,” IEEE Conference Publications, Pasadena, CA, USA, 2008.

[9] Boston Dynamics, “RHex - Devours Rough Terrain,” 2016. [Online]. Available: http://www.bostondynamics.com/robot_rhex.html. [Accessed 26 04 2017].

[10] Boston Dynamics, “YouTube,” 27 03 2012. [Online]. Available: https://www.youtube.com/watch?v=ISznqY3kESI. [Accessed 26 04 2017].

[11] S. M. LaValle and J. J. Kuffner, Jr, “Randomized Kinodynamic Planning,” The International Journal of Robotics Reserach, Iowa, USA, 2001.

[12] P. Dr. Corke, Robotics, Vision and Control, California: Springer, 2011.

[13] S. M. LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Iowa State University, USA.

[14] D. Belter and P. Skrzypczynski, “Integrated Motion Planning for Hexapod Robot Walking on Rough Terrain,” Institute of Control and Information Engineering, Poznan University of Technology, Italy, 2011.

[15] S. Bai and K. H. Low, “Terrain Evaluation and its Application to Path Planning for Walking Machines,” Nanyang Technological University, Singapore, 2001.

[16] Mark W. Spong, Seth Hutchinson and M. Vidyasagar, “Robot Modeling and Control,” Wiley, 2005.

[17] B. Jakimovski, Biologically Inspired Approaches for Locomotion, Anomaly Detection and Reconfiguration for Walking Robots, vol. 14, R. Dillmann, Y. Nakamura, S. Schaal and D. Vernon, Eds., Munchen, Germany: Springer International Publishing, 2011.

[18] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry and S. Schaal, “Fast, Robust Quadruped Locomotion Over Challenging Terrain,” IEEE Conference Publications, Alaska, USA, 2010.

[19] M. Bjelonic, T. Homberger, N. Kottege, P. Borges, M. Chli and P. Beckerle, “ETH Institutional Repository - Autonomous Navigation of Hexapod Robots With Vision-based Controller Adaptation,” 2017. [Online]. Available: http://e-collection.library.ethz.ch/eserv/eth:50586/eth-50586-01.pdf. [Accessed 29 04 2017].

[20] G. Carbone, Motion and Operation Planning of Robotic Systems, vol. 29, F. Gomez-Bravo, Ed., Cassino, Italy: Springer International Publishing, 2015.

[21] Ing. R. Woering, “Simulating the "First Steps" of a Walking Hexapod Robot,” Technische Universiteit Eindhoven, Eindhoven, 2011.

[22] A. R. Firas and Z. K. Hind, “Static Stability Analysis of Hexagonal Hexapod Robot for the Periodic Gaits,” University of Technology, Iraq, 2014.

[23] R. Campos, V. Matos and C. Santos, “Hexapod Locomotion: a Nonlinear Dynamical Systems Approach,” IEEE Press - University of Minho, Guimaraes, Portugal, 2010.

[24] J. Dr. Carrasco, Applied Mechanics for Industrial Robotics, Manchester: The University of Manchester, 2017.

[25] B. Siciliano and O. Khatib, Handbook of Robotics, Springer , 2007.

[26] D. How, “MIT - Aerospace Dynamics: Lagrange's Equations,” Massachusetts Institute of Technology, 2003.

[27] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, Robotics Modelling, Planning and Control, London: Springer-Verlag, 2010.

[28] D. GRZELCZYK, B. STAŃCZYK and J. AWREJCEWICZ, “Estimation of the Contact Forces Between the Hexapod Legs and the Ground During Walking in the Tripod Gait,” National Science Centre of Poland, Poland, 2016.

[29] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev and D. D. Lee, “Search-Based Planning for a Legged Robot Over Rough Terrain,” lEEE Conference Publications, Kobe, Japan, 2009.

[30] A. Elfes, “Using Occupancy Grids for Mobile Robot Perception and Navigation,” Carnegie Mellon University, 1989.

[31] D. S. Watson, EEEN30072 Mobile Robots and Autonomous Systems, Manchester: The University of Manchester, 2017.

[32] N. Amato, “Exact Cell Decomposition Methods,” Univ. of Padova, 2004.

[33] H. Choset, “Robotic Motion Planning: Cell Decompositions,” Carnegie Mellon University, 2014.

[34] C. Moskowitz, “Live Science - The Physics of Figure Skating,” 16 02 2010. [Online]. Available: http://www.livescience.com/6120-physics-figure-skating.html. [Accessed 13 04 2017].

[35] J. Lenarcic and B. Roth, Advances in Robot Kinematics: Mechanism and Motion, Netherlands: Springer International Publishing, 2006.

[36] M. B. Popovic, A. Goswami and H. Herr, “Carnegie Mellon University,” [Online]. Available: http://www.cs.cmu.edu/~cga/legs/implications.pdf. [Accessed 30 04 2017].

[37] R. M. Murray, Z. Li and S. S. Sastry, A Mathematical Introduction to Robotic Manipulation, San Francisco, Berkeley: CRC Press, 1993.

[38] W. Cheah, “FK and IK code,” Manchester, 2017.

[39] C. R. Ooi, “Balancing a Two-Wheeled Autonomous Robot,” The University of Western Australia, Nedlands, Australia, 2003.

[40] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi and M. Raibert, “Autonomous Navigation for BigDog,” IEEE Conference Publications, Anchorage, Alaska, USA, 2010.

[41] J. Ollervides, J. Orrante-Sakanassi and V. S. y. A. Dzul, “Navigation Control System of Walking Hexapod Robot,” Technological Institute of La Laguna, Torreón, Mexico, 2012.

[42] F. Tedeschi and G. Carbone, “Design Issues for Hexapod Walking Robots,” 2014.

[43] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Modelling and Control, John Wiley & Sons, INC..